1985

SILOUETT :

Carlos J. Sanchez
Lehigh University

SILOUETT

A Geometric Modelling System for IBM PC's

by

Carlos J. Sanchez

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Manufacturing Systems Engineering

Lehigh University

1985

Preface:

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

MAY 6, 1985

(date)

Tulga Ozsoy

Professor in Charge

Roger N Hoyt

Director of MSE Program

Eric L. Thompson

Chairman of CSEE Department

Contents

Contents

Contents

Contents

List of Illustrations

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to Dr. Tulga Ozsoy of Lehigh University's Mechanical Engineering and Mechanics Department for all his time, effort and guidance throughout my graduate work at Lehigh. Dr. Ozsoy's advise, encouragement and dedication had a significant effect on my thesis work.

I also would like to thank Mr. Ray Trombly and Mr. Joseph Chen of IBM Corporation at Boca Raton, Florida for there support and encouragement before and during my graduate studies at Lehigh.

It can not go without mention the work of Dr. Roger Nagel, Mr. Carlos Gomez and the rest of the staff at the Manufacturing Systems Engineering Department for helping us, the first graduating MSE class at Lehigh University, successfully complete our graduating requirements.

Finally I would like to dedicate the work done on this thesis to my daughter Melissa Mariel.

Abstract:

The beginnings of modern interactive computer graphics originate back in 1963 at MIT and the Ph.D. work on the Sketchpad drawing system by Dr. Ivan Southerland [SUTH63].

Computer Aided Design and Manufacturing (CAD/CAM) applications have been among the most dynamic and rapidly evolving within the computer graphics industry. To a large extent the growth of the computer graphics industry and that of CAD/CAM applications have been determined by the evolution of graphic display technology. The cost of computer graphics peripherals and the computationally intensive nature of CAD/CAM software, kept CAD/CAM applications captive to big mainframe computers for a long time. The increased capabilities of personal computers, the inexpensive cost of raster scan displays and solid state memory, have made possible the development of fairly sophisticated CAD/CAM applications for personal computers.

Silouett, an interactive Three Dimensional Geometric Modeling System (3D Wire Frame) has been developed

for the IBM family of personal computers. Besides the obvious advantage of true 3-D capability, which allows for the proper viewing of tree dimensional objects, the graphics software is based on GKS, a device independent graphics standard. To further promote portability between systems, the program modules were written, to a large extent, in standard Pascal. Finally, Silouett's architecture features a Network data base system and a Draw File concept that greatly enhance the speed with which graphic operations can be performed.

Chapter 1 - Introduction:

This thesis presents the process followed by the author in developing a 3D geometric modeling system for the IBM Personal Computer. To implement all the features usually found in a commercially available geometric modeling system would be outside the scope of a Masters thesis and impossible to do within the 18 months spent by the author at Lehigh University. Therefore, the approach taken was to develop an architecture based on that described in [OZSO83] that allows the system to be easily expanded in the future. The software developed provides the foundations for the system.

Until recently the computationally intensive nature of geometric modeling systems kept them constrained to mainframes and powerful mini computers. Recent advancement in the personal computer industry has trigger and interest in developing advance engineering software for personal computers. Geometric modeling is one the areas that has experience rapid growth in the personal computer market. However, the computationally intensive nature of geometric modeling is still present and

with personal computers more than with mainframes or minis, the developer must be aware and take advantage of all features that could possibly reduce the computational load. In developing Silouett, the geometric modeling system described in this thesis, the author has tried to optimize the computational aspects of the system and carefully analyzed the database structure in an effort to keep the system response time within an acceptable level.

Chapter 2 presents an overview of the techniques required to manipulate three dimensional objects in space and how to display them on an output device such as a raster scan monitor or an XY plotter. These techniques represent one of the foundations upon which the geometric modeling system is built.

Once the fundamentals of viewing an object in three dimensions were understood, we turned our attention to the internal representation of the object itself. Indeed a significant part of the effort to develop Silouett was spent in developing an efficient data base system. A network database system with its own dynamic storage facility was developed as part of this thesis work. Chapter 3 documents the issues the

author confronted in developing the data base system.

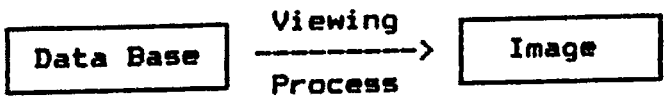
With the two building blocks of the geometric modeling system understood, our attention was concentrated on the programming aspects of the geometric modeling system. To allow the system to be easily expandable in the future and to keep reasonable control over the system, the software was developed in a highly structural manner. Chapter 4 describes the major software modules that make up the system and how they interact with each other.

As implemented today, the system required approximately 8000 lines of PASCAL code not including the device drivers which are commercially available. It requires 134KB to load and a minimum of 64KB of additional dynamic storage to run. Therefore a system configured with at least 256KB and the required graphics card will be able to run Silouett.

Chapter 2 - Viewing in Three Dimensions:

Fundamentally, the images produced by a geometrical modeling system, can be considered a collection of graphic and text entities. Assuming that this collection of entities is available to the system in the form of its data base, we can define the viewing process.

The objective of the viewing process is to transform the information contained in the data base into an image that can be displayed in some sort of output device such as a graphics monitor.



The viewing process is a complex process that can be subdivided into four main processes :

1. Coordinate and Image Transformations
2. Clipping
3. Mapping
4. Drawing

The objective of this chapter is to progressively develop a background in the viewing process so that the programming considerations chapter can be better understood. The mathematical elements related to the viewing process have been extensively documented by researchers in the field. Therefore no attempt is made to duplicate this work, rather, a brief background and references to a more comprehensive discussion are provided.

2.1 Coordinate Transformations:

The user of a geometric modeling system, must describe in geometric terms, the object or part of the world the user wants to model. To accomplish this task, the geometric modeling system provides a set of graphic primitives such as points, lines, arcs, etc... to describe the basic geometric attributes of the object.

As users of a geometric modeling system, we would like to be able to define our model in reference to a preferred coordinate system (Polar, Cylindrical, Spherical etc) that best relates to whatever is we are modeling. We refer to this frame of reference as

the User Coordinate System, UCS. However this may or may not be the best coordinate system to store the geometrical information.

In order to satisfy both the user and the requirements of the geometric modeling system, Silouett supports five reference coordinate systems:

1. Absolute Coordinate System (ACS).
2. Global Coordinate System (GCS).
3. Work Coordinate System (WCS).
4. Normalized Device Coordinate system (NDC).
5. Device Coordinate System (DCS).

Both the NDC and DCS are only used internally by Silouett and thus are transparent to the user.

The user is most likely to use the ACS and WCS coordinate systems to describe objects, however some operations allow the user to use the GCS to describe them.

2.1.1 Absolute Coordinate System:

The Absolute Coordinate System is a right handed 3D

cartesian coordinate system used to describe the object being modeled . The ACS is attached to the object and remains fixed in space with respect to it. If the object was previously defined using a non-cartesian coordinate system, the user is responsible for mapping geometric information from the UCS to the ACS before the object is described to Silouett.

The ACS is the coordinate system used to store geometric information in the data base. In Silouett, the resolution and range of the ACS is only limited by the internal representations of real values in the IBM PC. The units of the ACS are specified by the user.

2.1.2 Global Coordinate System:

The Global Coordinate System is a right handed 3D cartesian coordinate system fixed to the lower left corner of the screen (or output device).

The main purpose of the GCS is to allow the objects to be displayed with the appropriate scale factors. Since the physical dimensions of the output device

are know to Silouett, objects can be displayed at whatever scale the user specifies.

The resolution of the GCS is the same as that of the ACS. However the units are always millimeters irregardless of the units used in the ACS.

2.1.3 Work Coordinate System:

At times, it would be desirable to describe parts of the object using a local coordinate system rather than the ACS. The work coordinate system, WCS, is such a reference coordinate frame.

Silouett supports WCS as a graphic entity, thus the user is free to define as many WCS as necessary to simplify the task of describing the geometry of the object. The resolution, range and units of the WCS are the same as those of the ACS.

2.1.4 Normalized Device Coordinate System:

The Normalized Device Coordinate system is a coordinate system used to provide a device independent means of representing display data.

Silouett uses the NDC as a means to communicate with the specific device drivers supported by Silouett. Since all devices supported by Silouett have a two dimensional view surface, the NDC is a 2D coordinate system. The range of the NDC system extends from 0 to 32767, thus allowing the use of integer arithmetic to represent NDC coordinates.

2.1.5 Device Coordinate System:

The Device Coordinate System is a device specific coordinate system which defines the resolution of the output device.

A device driver provides the necessary mapping between data represented in NDC and the DCS required by the specific output device. It is also responsible for interpreting graphics commands and instructing the hardware on how to accomplish the required task. A device driver is required for each input or output device supported by Silouett.

2.1.6 Graphical Kernel System:

Graphical Kernel System (GKS), is a graphical system which allows application programs to support a wide variety of graphic devices. It is defined independent of programming languages and concerns itself with both input and output. It was accepted as a Draft International Standard in 1982 by the International Organization for Standards (ISO).

Silouett makes use of a GKS graphics package to transformation from NDC to DCS and drive the output devices.

2.2 Image Transformations:

The ability to display and manipulate an object in three dimensional space is fundamental to the understanding of the shape of that object. When we are presented with an object foreign to us, we tend to pick it up and look at it from different angles in order to obtain a better understanding of the object. For the same reasons, when working with geometric models, we want to be able to manipulate the object in space. The transformations that allow

us to scale, rotate, translate and reflect the image of the model are called Image transformations.

The mathematical elements dealing with such transformations have been well documented by several researchers of the computer graphics field [ROGE76], therefore only minimum background will be provided here.

As Figure 1 on page 11 illustrates, in the process of applying Image transformations, we also map the ACS to the GCS>

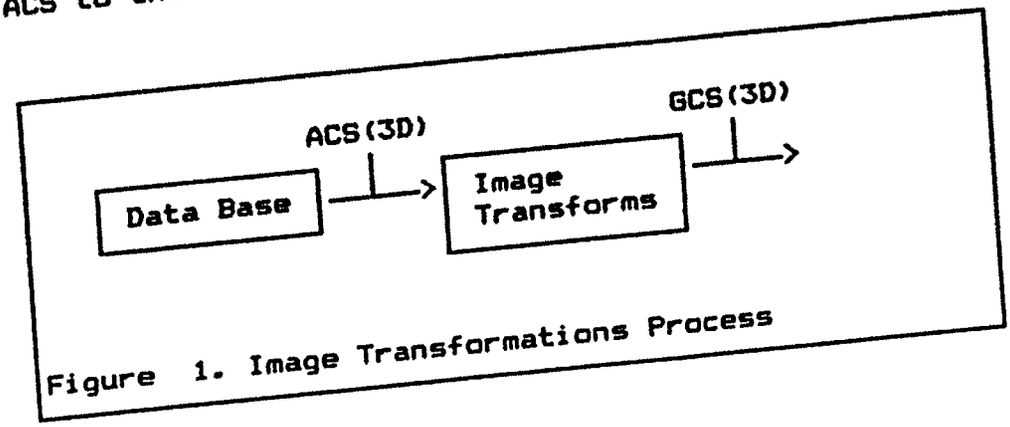


Figure 1. Image Transformations Process

When the transformations are applied to the object model, the model and not only its image is change. In this case we call the transformations Object Transformations.

In this chapter, when we refer to the object, we are referring to the image of the object rather than the object model itself. However the discussion of image transformations applies equally as well to object transformations.

2.2.1 Homogeneous Coordinates:

Homogeneous coordinates have been used extensively in the geometric modeling. In general, the homogeneous coordinates for an N dimensional space, are represented by N + 1 coordinates. Thus, in three dimensional space, points are represented as P[X,Y,Z,H], were H is any non-zero scale factor usually taken as H=1 [ROBE65].

The introduction of homogeneous coordinates allow us to define a general 4 x 4 transformation matrix, GTM, of the form :

$$GTM = \left[\begin{array}{ccc|c} R11 & R12 & R13 & Px \\ R21 & R22 & R23 & Py \\ R31 & R32 & R33 & Pz \\ \hline Tx & Ty & Tz & S \end{array}\right]$$

Where

R is a 3 x 3 Linear Transformation matrix that produces :

- Local Scaling
- Reflections
- Rotations

T is a 1 x 3 Row Matrix that produces translations.

P is a 3 x 1 Column Matrix that produces projections.

S produces Overall Scaling

The general transformation matrix allows us to apply all of the above mentioned transformations at once.

2.2.2 Coordinate Frames:

In Silouett, each view of the model has a GTM associated with it. If we want to apply a second transformation to modify the view, the order in which the transformation matrix is applied determines the coordinate reference frame with respect to which the transformation takes place. If we premultiply the view GTM by a second transformation matrix, the transformation takes place with respect to the ACS, if we postmultiply, the transformation is with respect to the GCS.

2.2.3 Concatenation of Transformations:

Transformation matrices can be concatenated in the right order and then applied to the model. If we are applying the transformations to a single point, it makes no difference to apply each transformation one at a time or to apply the concatenated transformation once. However, if we are dealing with a fairly complex model, composed of many points, the concatenation of the individual GTM's into a complex GTM drastically reduces the number of multiplications required to accomplish the transformation. Thus reducing the time it takes to complete the task.

2.2.4 Change of Coordinate Systems:

An alternative but equivalent way of thinking of a transformation is as a change of coordinate system [FOLE82]. Therefore, if we want to apply a series of transformations with respect to a local coordinate system, we first apply a transformation that will align the WCS with the reference coordinate system, then apply the transformations with respect to the WCS, and finally apply a transformation that takes the WCS back to its original position.

2.2.5 Inverse Transformations:

We have seen that the transformation of coordinate systems requires the use of inverse transformations, in a similar manner the need for inverse transformations arises when we want to cancel the effects of a transformation just performed. Also when using an interactive input device, such as a graphics tablet or tracking cross, inverse transformations are used to locate or define entities.

The task of obtaining the inverse of a transform can be simplified by realizing that the 3x3 R submatrix of the GTM is an orthogonal matrix, and thus the inverse of R can be found by transposing.

2.2.6 Projections:

In general, projections transform points in a coordinate system of N dimensions to a coordinate system of dimension less than N. In our case, we are interested in projecting the 3D data defining the object to a 2D plane so that it could be displayed in

an output device. This type of projection, know as planar geometric projection, can be divided into two basic classes :

1. Perspective Projections - The center of projection is at a finite distance from the projection plane.
2. Parallel Projections - The center of projection is at infinity.

Projections are be defined by the 3 x 1 column submatrix T of the general transformation matrix. A detailed description of the mathematics involved in the projection process is presented by [ROGE76] and [FOLE82].

2.3 View Manipulations:

In a geometrical modelling system, a View is defined as the projection of the Image of an object onto the surface of an output device. So far we have discussed but one of the processes that affect such a projection, the Image Transformation. This section describes the remaining processes that form part of

the viewing process.

2.3.1 Viewing Volumes and Clipping:

We indicated before that in the process of transforming the image of an object, we also change coordinate systems, going from the ACS to the GCS. Contrary to the ACS, which is essentially unbounded (see ACS definition), the GCS is bounded by the physical dimensions of the output device. Therefore we must tell the geometrical modelling system which portion of the GCS contains the graphic information to be displayed at this time. We do this by specifying a viewing volume in the GCS.

The viewing volume can be defined by specifying the coordinates of the main diagonal of a cube. In the case of Silouett, the sides of the cube must be parallel to the GCS axis.

Although the view volume X and Y coordinates can not exceed the physical dimensions of the output device, it need not coincide with the bounds of the GCS. As we shall see, this will only result in the image being mapped to a particular region on the output

device.

Once the viewing volume has been defined, the geometrical modelling system must insure that no graphic entity has GCS coordinates that exceeds the bounds of the viewing volume. If the coordinates of a graphic entity exceed these bounds, the geometrical modeling system will "clip" them so that they will lie either entirely within the viewing volume or entirely outside of it. This process is know as clipping [FOLE82].

Note that once the graphic entities have been clipped to the viewing volume, the Z coordinate is not needed in order to map the coordinates to the output device. However, the Z coordinate may be retained for other purposes, controlling the brightness of displayed points for example.

The combination of image transformation and clipping is know as windowing, Figure 2 on page 19 illustrates this process. In 2D, the view volume is not a volume but an area called a window, thus the term windowing.

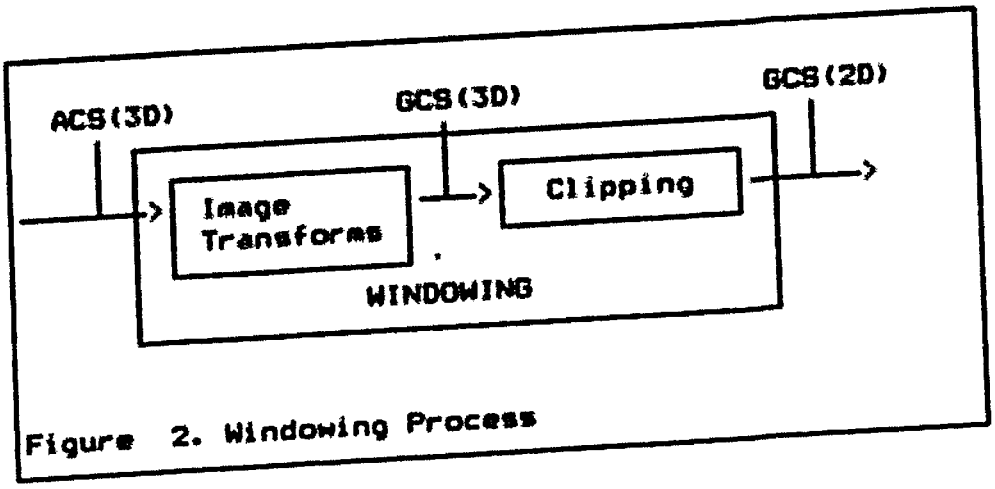


Figure 2. Windowing Process

2.3.2 Viewports and Mapping:

A viewport is a rectangular section of the output device onto which the view volume and thus its contents are mapped.

At this time we could map the view volume contents directly to the output device. However, my mapping to a normal device we introduced data independence from the output devices supported by the geometric modeling system. A normalized device, is a logical device whose view surface is square and NDC coordinates range from 0 to 32767.

By mapping to a normal device first, we can redraw the entire view in any other device without the need

to go through the windowing process again. On doing so, the original scale of the drawing will not be preserved but the aspect ratio will (provided the device viewport is square too). If we want to preserve the original scale, an additional clipping operation must be performed.

Even if we have to perform an additional clipping operation, the use of the normal device concept is useful. Note that this additional clipping can be done on NDC and is done only on the portion of the model that was visible in the view volume. Thus the time required to do the second clipping is significantly less than it would be required if the view was generated from the database again.

Silouett implementation of Normal Devices varies somewhat from the one previously given because of performance considerations. A complete discussion of the subject is included in the programming considerations chapter. Figure 3 on page 21 illustrates the mapping process.

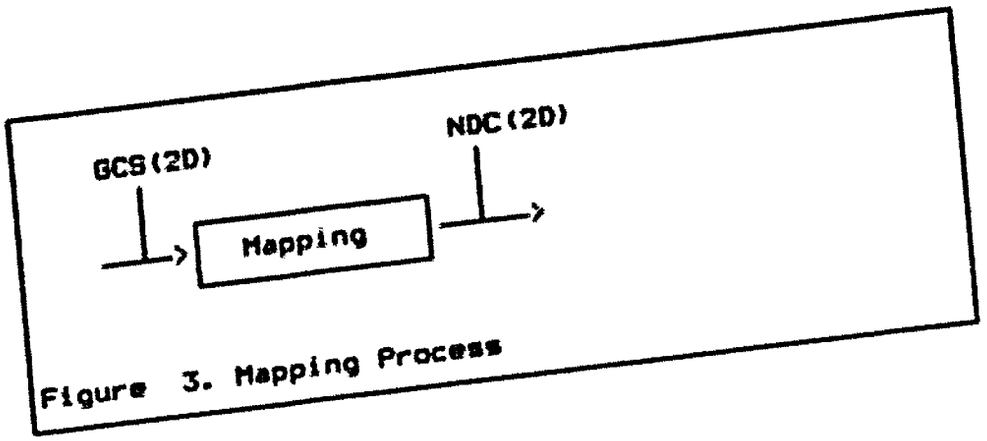


Figure 3. Mapping Process

2.3.3 Drawing:

Mapping to the output device completes the viewing process, Figure 4 on page 22. This is a device specific operation which is usually handled by a device driver.

The device driver converts the normal device coordinates to device coordinates, interprets, and executes basic graphics commands (Draw Point, Draw Line, etc) .

How the entire viewing process is accomplished in Silouett is discussed in the programming considerations chapter.

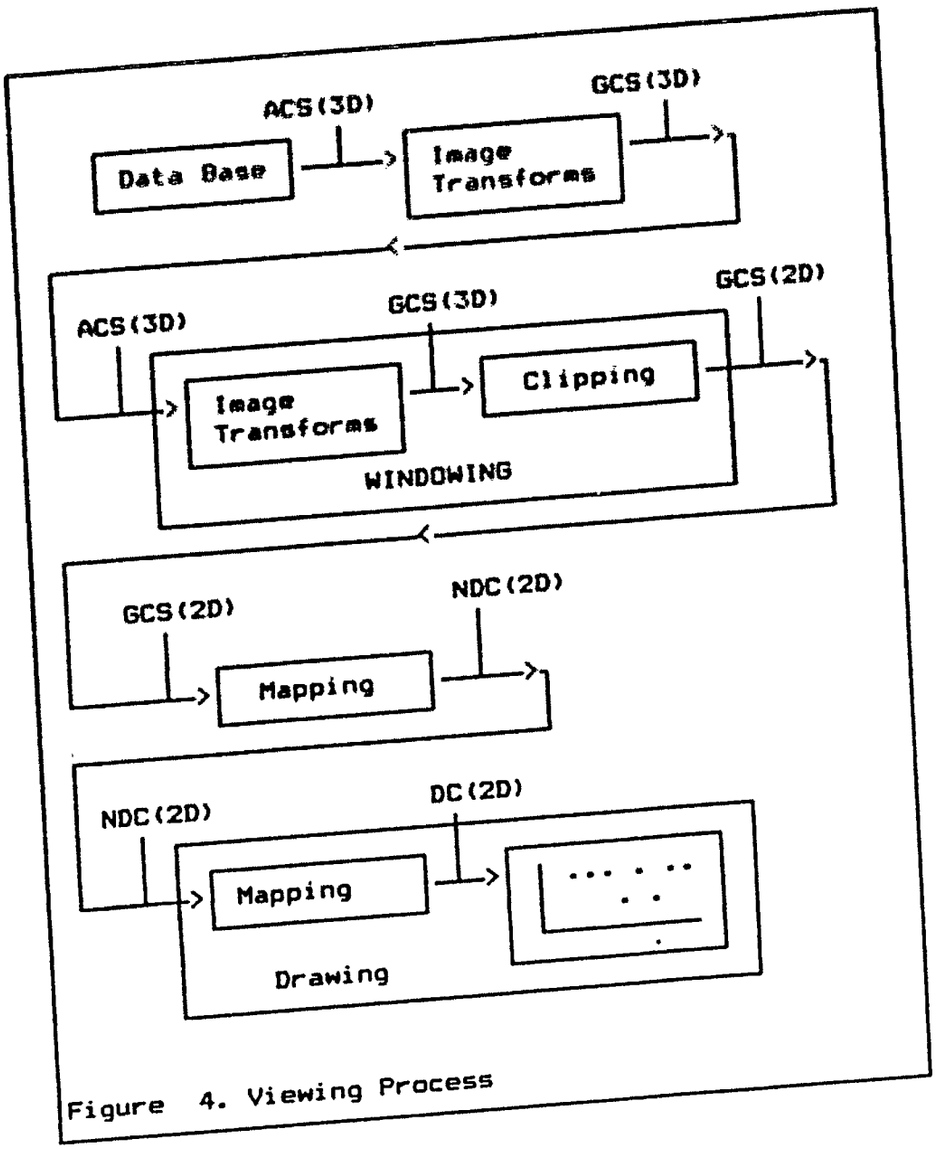
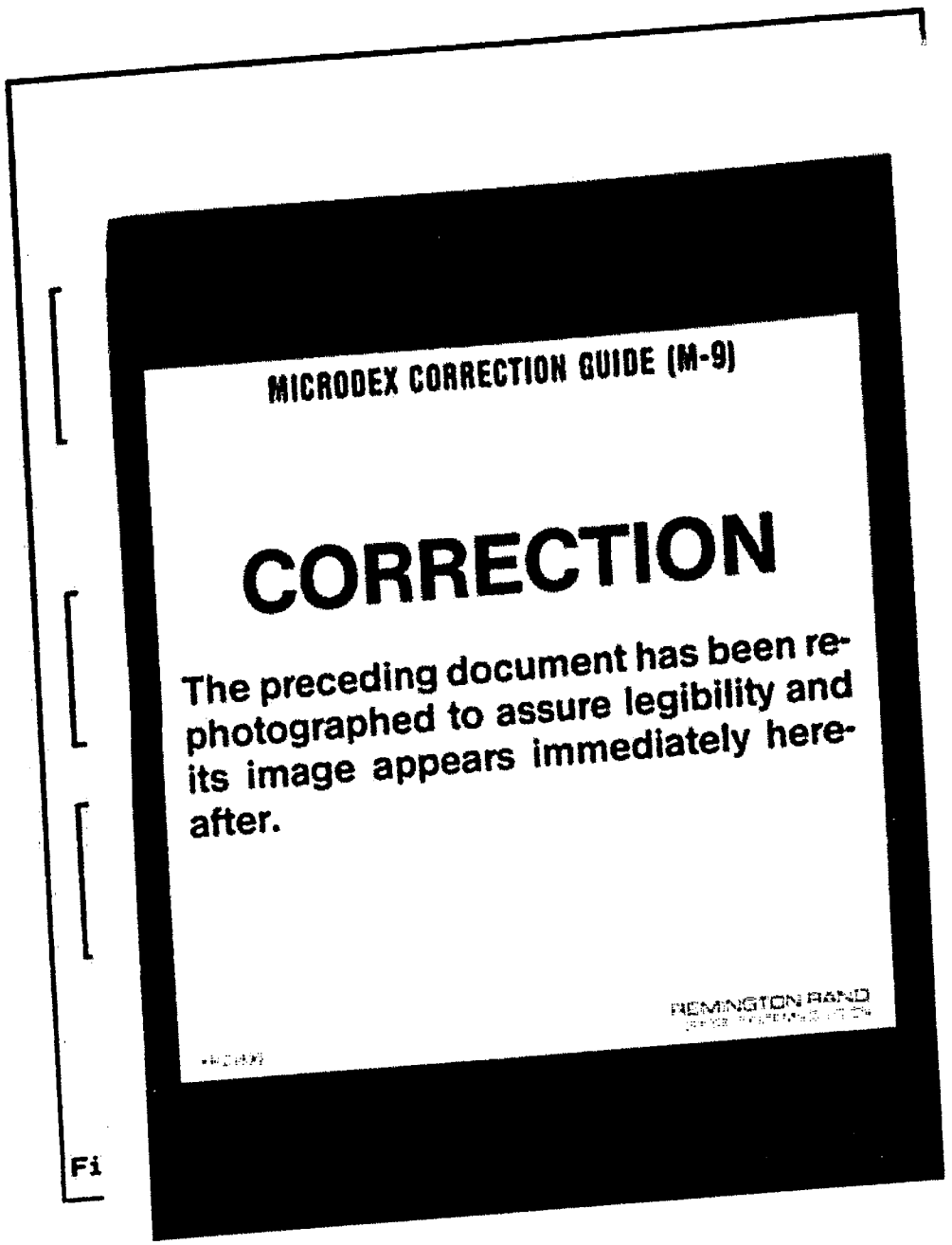


Figure 4. Viewing Process


MICRODEX CORRECTION GUIDE (M-9)
CORRECTION
The preceding document has been re-
photographed to assure legibility and
its image appears immediately here-
after.
REMINGTON RAND
Fi

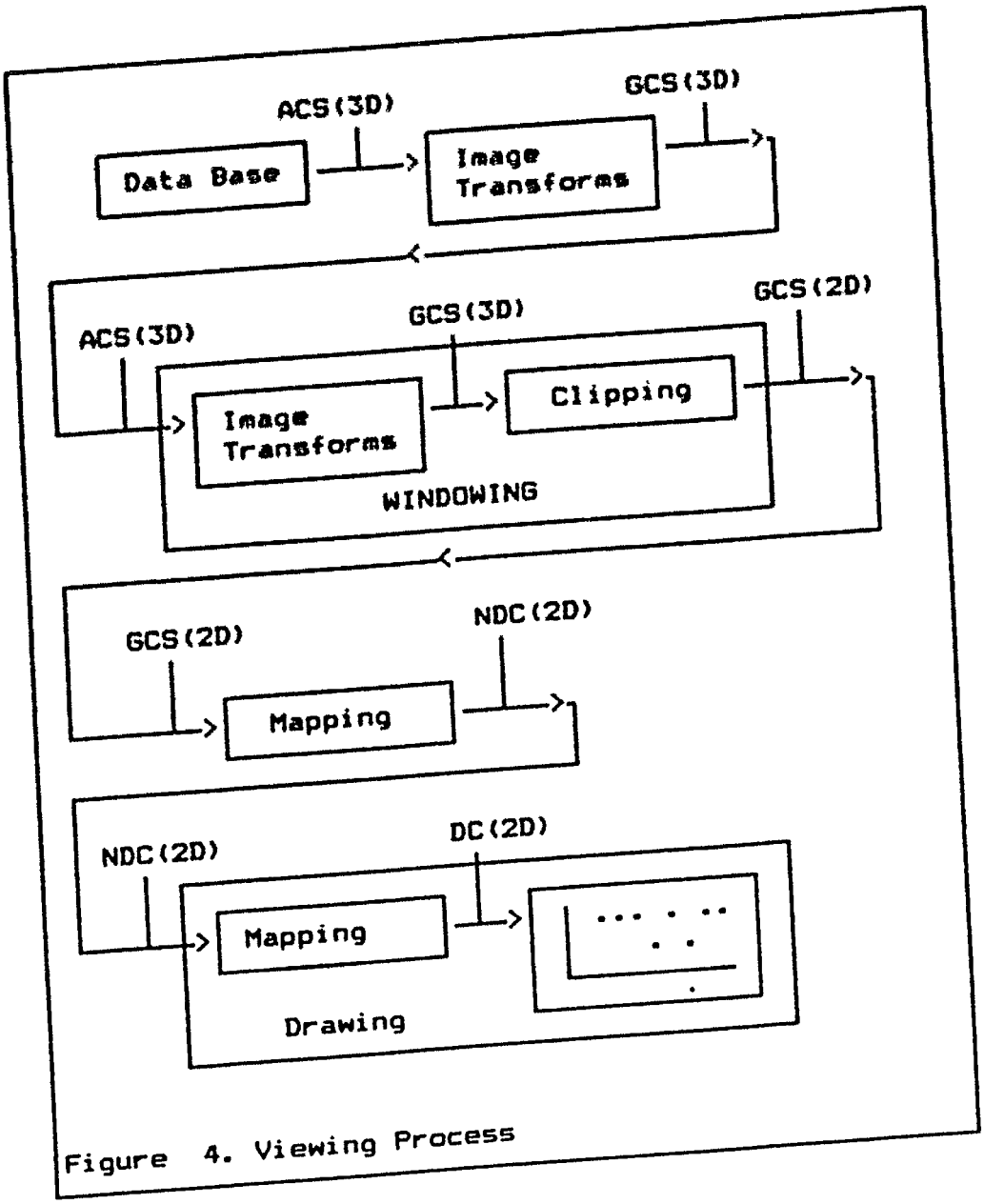


Figure 4. Viewing Process

Chapter 3 - Data Base:

Wolfgang K. Giloi [GILO78] defines computer graphics as:

> The art of representing graphic objects in a computer so that they can be visualized in pictorial form, of providing an interactive mode of operation in order to generate and manipulate such objects, and of associating graphic objects with other pertinent data to the application program.

Thus, one of the foundations of a geometric modelling system is its data base. To a large extent the efficiency and flexibility of the system depends in turn on the efficiency and flexibility of its data base and data management routines.

The user of the geometric modelling system, must describe in geometric terms, that portion of the world the user wants to model and produce an image off. To understand how the geometric modelling system can keep track of this information, we must

agree on a set of terms to describe data and its representation.

Entity An entity is a graphic element about which information is recorded in the data base.

Attributes Every entity has some basic attributes that characterize it. Attributes provide a means of differentiating members of the same entity group.

Data Value A data value is the actual data or information contained in each data element. The values taken by data elements can be quantitative, qualitative or Boolean depending on how the data element is used in the data base.

Data Record A data record is a collection of values taken by related data elements. Figure 5 on page 25 shows the values taken by the attributes of POINT, ie... 10, Blue, Cross, Visible, etc, collectively they constitute a data

record.

Entity	Attributes (Data Element)	Values (Data)
POINT	ID Label	10
	Color	Blue
	Symbol	Cross
	State	Visible
	X	5.00
	Y	10.00
	Z	0.0
LINE	ID Label	5
	End PNT1	12
	End PNT2	16
	Color	Yellow
	Density	2
	State	Invisible

Figure 5. Entities, Attributes and Values

Data File — A data file is a collection of homogeneous or heterogeneous data records.

Data Base — A data base is a collection of related data about an enterprise with multiple uses. Thus, a graphics data base is simply a collection of related data (graphic entities) shared by the application programs.

DBMS — A Data Base Management System (DBMS) is needed to integrate the data files into a data base and supervises all data file manipulations. The software and procedures that manage the data base comprise a Data Base Management System.

3.1 Data Base Design:

The development of a graphics data base can be

divided into tree related tasks [ATREBO]:

1. Development of a Conceptual Data Model
2. Defining the Logical Data Model
3. Physical Implementation

3.1.1 Data Independence:

Data independence is not a programming technique but a programming discipline. System parameters can be pass to the geometric modelling system at execution time to provide the user with some flexibility in customizing the system. An efficient and simple way to accomplish this is through the use of a Profile File.

A Profile File is a dataset accessible to the user and processed by the system at execution time, in which system constants and global variables are defined and values assigned to them.

Another aspect of data independence has to do with the internal representation of the data. A thorough analysis of the entities and their relationships

minimizes the impact of their representations in the application programs that use them.

3.1.2 Data Dictionary:

A Data Dictionary is a central document in which information about the entities, data elements and relationships between entities, their origins, meanings, uses and representation formats are collected.

As the data base expands, new applications are developed and integrated, the data dictionary is updated to reflect the latest changes.

3.1.3 Data Models:

Before we can proceed with the Logical Data Model phase, we must decide on an underlying structure for our data model. Many factors need be considered, the following constitute the very basic ones:

1. The DBMS must make reasonably efficient use of the available or projected computing resources.

2. The data base must be able to provide the necessary information in an acceptable period of time. This requirement is more stringent in the case of interactive system where the users expect an almost immediate response to their queries.

3. The data base must be capable of being updated and expanded without too much difficulty. Provisions for adding, deleting and modifying data entities in a timely and efficient manner should be provided.

Data base designers can choose from several data base models. Today most data base implementations use relational, hierarchical or network data models [Appendix C]. The differences in the models involve the way in which data are organized, which not necessarily bear any relation to the way in which the data are physically stored in the computer.

Although in theory the most desirable data base is relational, Silouett makes use of a network data base. At the expense of a much more complex data

structure, the network data base offers a significant performance improvement over relational data bases today. This is an advantage that can not be ignored when we consider the performance of personal computers and the computationally intensive nature of geometric modelling systems.

3.2 Data Structures and Representations:

Any representation of a data structure in the computer memory must encompass the data to be stored as well as the implicitly given relationships which constitute the structuring of the data [ELS075]:

Data Representation { Data structure relationships
                    { Data element (data value)

There are two basic types of computer representations of data structures:

1. Location Addressed
2. Content Addressed

In either case we must provide an addressing function that maps the logical memory structure to physical memory. We can define two basic types of addressing

functions:

1. Sequential allocation - data are stored in consecutive memory locations.

2. Linked allocation - allows an arbitrary order of the data within memory by specifying data element successor and predecessor relationships, ie... addresses or pointers are stored within the data records.

Silouett makes extensive use of linked data structures throughout its data base.

3.2.1 Sequential Data Files:

The most elementary data organization is the sequential data file. It consist of a directory file that specifies the beginning and ending addresses of a file of records and a data vector that contains the data records themselves. Figure 6 on page 32 illustrates this concept.

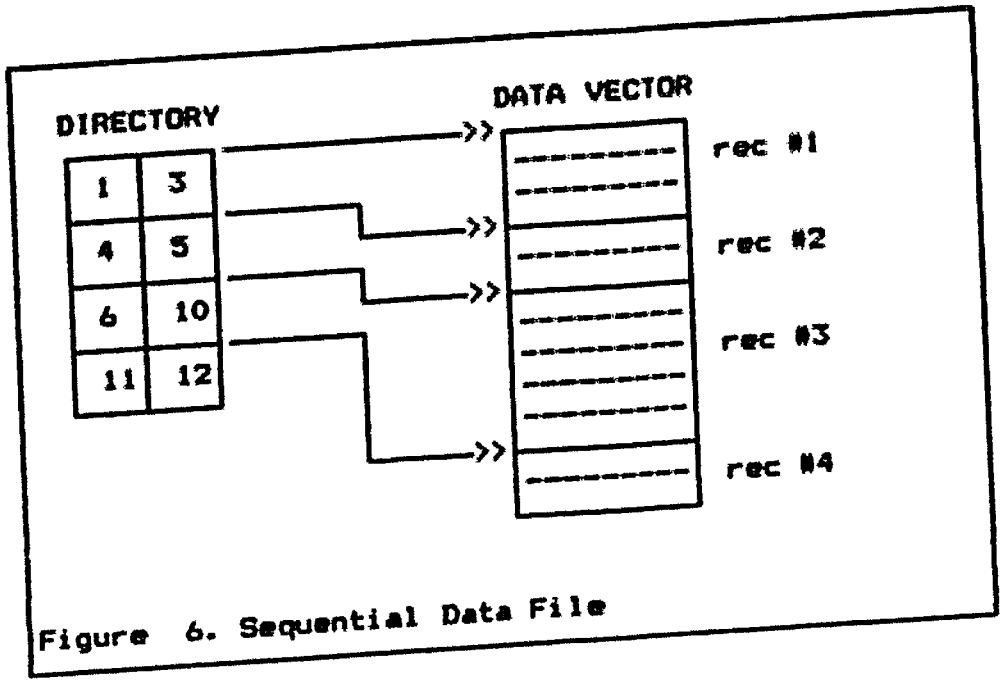


Figure 6. Sequential Data File

3.2.2 Link Lists:

Link lists although more complex than sequential files, allow much more flexibility. Data structures are defined by a constant logical scheme, but within the confines of the scheme, possibilities for reorganizing the data and changing the element interrelationship will be practically unlimited. Figure 7 on page 33 graphically illustrates a record node, the basic building block of any link list.

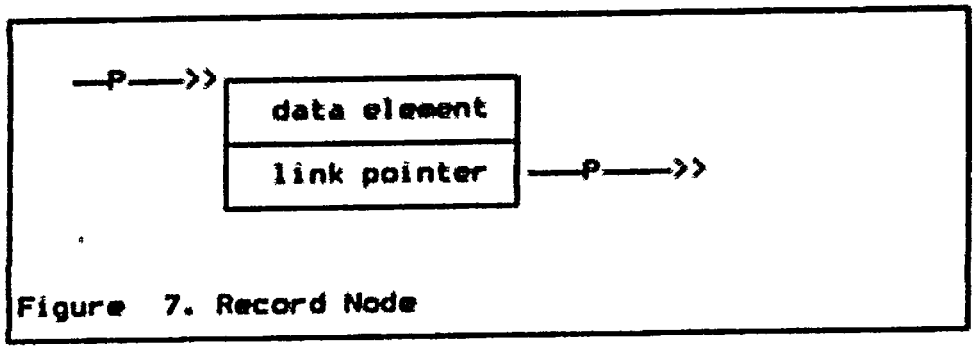


Figure 7. Record Node

The forward link list is the simplest of all dynamic data structures from which all others evolve.

- The first element of the list is a list header that identifies the list.
- Ordering is represented explicitly within the elements of the data record. Each node contains a link to the next node in the list.
- The last element of the list has a null link entry to identify the end of the list.

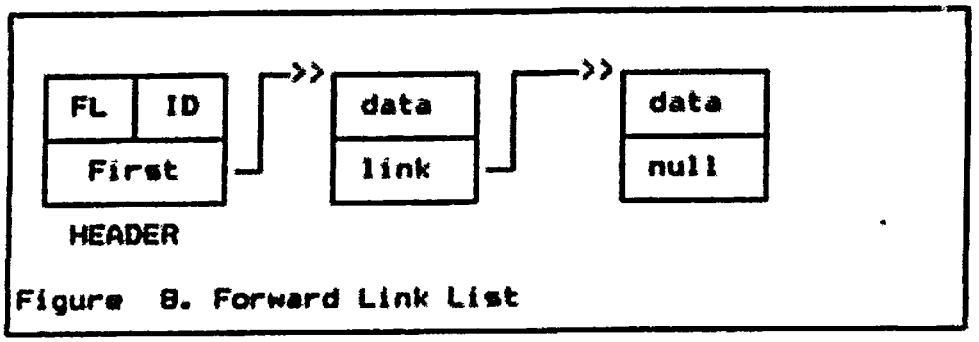


Figure 8. Forward Link List

Because of the way that lists are constructed, the user is restricted from directly addressing the header, and therefore it would be necessary to have special code to handle the first record on the list. To avoid this the concept of sentinels is introduced. A sentinel is basically a dummy entry of the same format as any other record node. It guarantees the list will never be empty, as by definition the sentinel can never be deleted or moved during the life of the list. It can however be directly addressed by the user as any other record in the list, sentinels eliminate the need for especial code to handle the first record in the list.

There are several differences between linked and sequential allocation [KNUT68]:

1. Linked allocation takes up additional memory space for the links, however, the improved

performance obtainable from it more than compensates for the storage inefficiency.

2. It is easy to delete nodes from linked lists.

3. It is easy to insert nodes anywhere in the linked list.

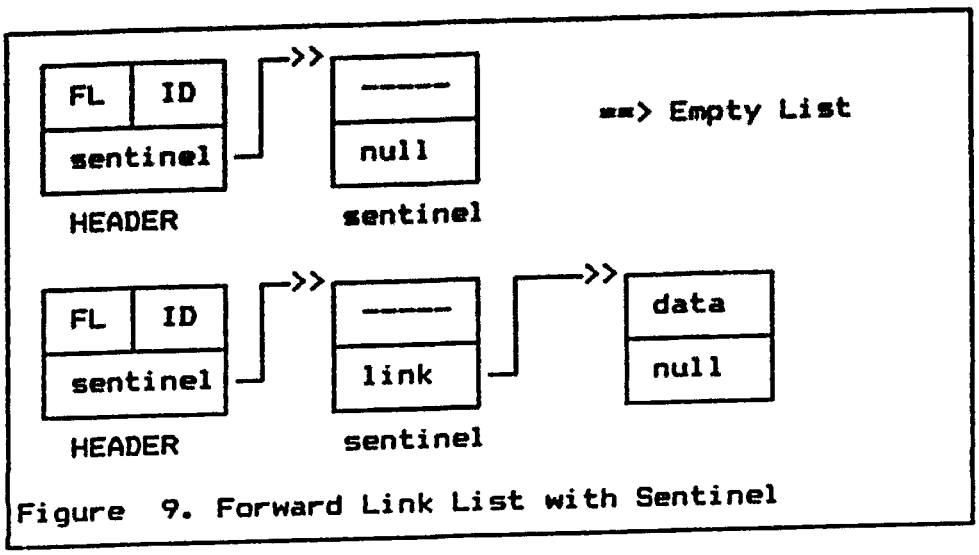


Figure 9. Forward Link List with Sentinel

Next in complexity to the forward link list is the circular list or ring. It is a forward link list in which the link field of the last element is not null but rather it points back to the list sentinel.

- The entire list can be traversed starting from any node.
- By flagging the sentinel, it is possible, given any list node, to find the sentinel and thus the beginning of the list.

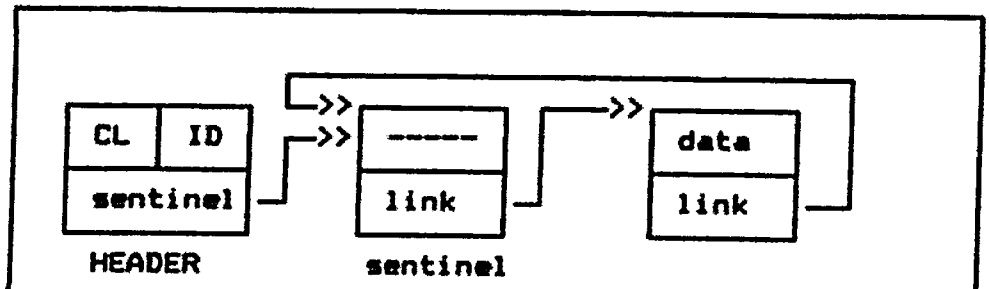


Figure 10. Circular List with Sentinel

By including two link fields in each node, a forward link and a backward link, we can obtain even greater flexibility in the manipulation of linear and circular lists.

Besides the obvious advantage of going back and forth on the list, the process of deleting a node from the list is greatly simplified. In a single link list, we must keep track of the predecessor node, as its link field must be changed, a requirement lifted by the double linked list.

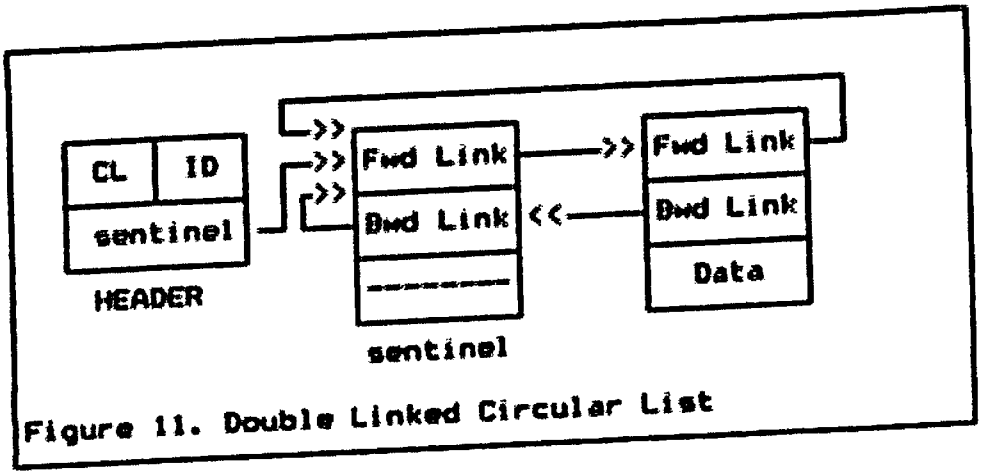


Figure 11. Double Linked Circular List

3.3 Storage Management

So far we have considered the logical characteristics of data structures and its physical representations. A third area we must consider is the relation of the various structures and the computer storage environment in which they reside.

Some computer languages, such as BASIC and Pascal, support dynamic allocation functions that manage storage space for us. Others, like FORTRAN, don't support dynamic allocation, therefore the applications programmer must provide its own storage management environment. In either case, both physical storage and processing time are finite resources that must be manage as efficiently as

possible. It is not uncommon for data base programs to run out of storage space or encounter prohibitive processing times.

The environment of the application determines were the responsibility for storage management lies. Before we discuss Silouett's implementation, it will be helpful to look back at some of the issues associated with storage management.

3.3.1 Free Storage Lists:

Initially, free storage lists are likely to consists of a single large block of storage reserved for the purpose of dynamic allocation. As blocks of storage are allocated and subsequently freed, fragmentation of the free storage block occurs, therefore free storage is more generally represented by a list of free storage blocks. Each node in the list requires a link field, pointing to the next node, and a size field to indicate the block size.

Typically, free lists are organized as single linked lists with a sentinel node. The header node, has a

pointer, First_Free, to the sentinel node. When a request is made for a block of n units, the last n units of the initial storage block are allocated and a pointer to the beginning of the n units block is returned. The size field of the original block is adjusted to reflect the allocation. When storage is returned, the block is inserted as the first element of the free storage list. The process continues until free storage is exhausted, in which case no further storage requests can be granted until some storage is freed. If the free list is not exhausted, but the requested block is larger than the largest available one, the request is also denied.

3.3.2 Allocation Techniques:

As described above, the free list is ordered solely on the basis of time of entry into the list. Ordering is completely independent of any characteristics of the block, such as size or location. This technique, although the fastest possible for inserting blocks into the list, may not be the optimal one for our application. Consider next the methods available to satisfy a storage request.

1. First Fit Method - The technique is simply that of finding the first block large enough to fulfill the request.

2. Best Fit Method - If we believe that it is unwise to break up a large block when a smaller one will do, we may want to search for the smallest block that will fulfill the request.

3. Worst Fit Method - Finally, we may want to fulfill the request using the largest block available.

Which method is best depends on the application's environment. The best fit method tends to create a wide spectrum of free block sizes. The worst fit method, always subdivides the largest block so that all blocks tend to be of approximately the same size.The first fit method, selected for Silouett's implementation, tends to be intermediate in its effects because of its essentially random selection technique.

3.3.3 Storage Compaction:

Finally we must recognize that sometime after several allocations and and storage returns have been processed, we can end up with contiguous blocks of storage that the system treats as independent not contiguous blocks. It could happen that the storage management routines will deny a request for a large block on the basis of not finding a large enough block. However, if two or more contiguous blocks could be collapse into one block, the request could have been granted.

What kind of intelligence is needed to recognize contiguous blocks and collapse them into one ?
We must, whenever a block is freed, check its starting location against the ending location of every block on the list, and in cases were the freed block has a lower location we must also check its end against the start of every block. This obviously will create a huge overhead for freeing a block. However, the problem leads to an alternative way of ordering storage blocks, namely by location or relative address. The use of this technique will result in an average search of length N/2 to return a

block to free storage, but it only requires checking the address of the successor and predecessor nodes in other to decide if the blocks can be collapsed. Further more, the use of this technique virtually eliminates the need for a method to recover unused storage space. Two techniques available for recovering storage space are Garbage Collection and Reference Counters [ELSO75].

3.4 Silouett's Data Base Implementation:

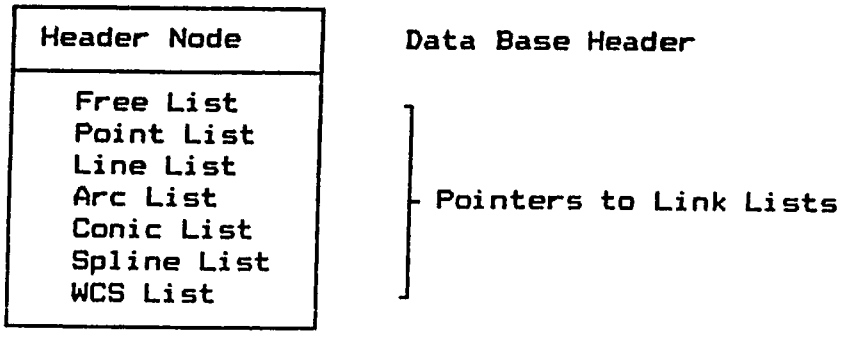
As we mention before, Silouett implements a network data base. Of paramount importance in the decision to implement a network data base structure were the performance efficiency that can be obtained by predefining data relationships, and the ability of the model to represent many-to-many relationships [Appendix C].

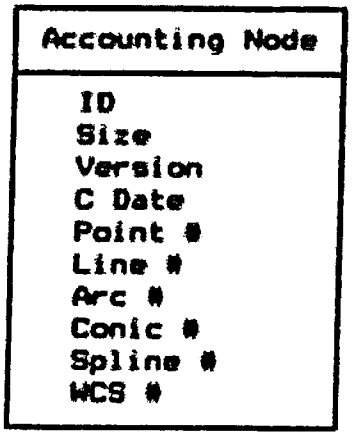
Silouett's data base software module can be subdivided into three major areas.

1. Entity and Data Definitions
2. Storage Management
3. List Management

3.4.1 Entity and Data Definitions:

At this time, Silouett's data base supports Points, Lines, Arcs, Conics (Ellipse, Parabola and Hyperbola), Cubic Splines and Work Coordinate System entities. The structure of Silouett's graphic entities is similar to that described in [OZS084] and was intended to be compatible with it to some extent. However, there are significant differences as Silouett first does not support surfaces and second implements the data base using a different data base model. These entities are maintained within the data base as double-linked lists to facilitate data base management operations. Besides entities, data base global and control variables are also maintained within the data base.

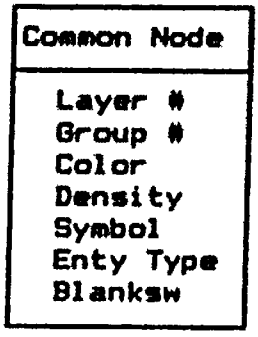




Maintains Data base status

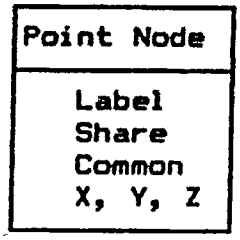
... Data base id
size
version #
Creation date

Counters to keep track of number of entities in data base



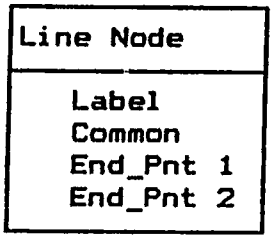
Information common to all entities

... Layer # of entity
Group # "
Color "
Density "
Symbol "
Entity type
Visibility switch



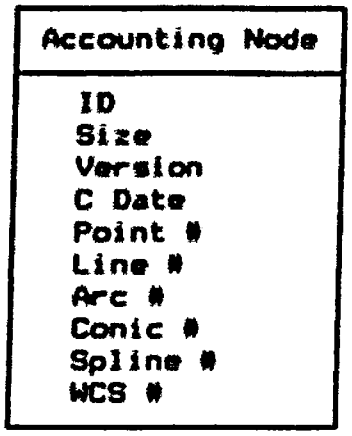
Defines a Point entity

... Point id
Share counter
Common information
Coordinates of point (ACS)



Defines a Line entity

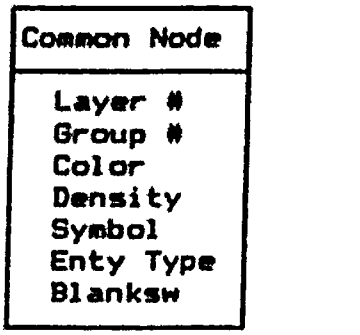
... Line id
Common information
Pointer to end point
Pointer to end point



Maintains Data base status

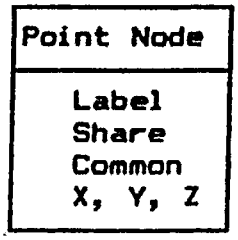
... Data base id
size
version #
Creation date

Counters to keep track of number of entities in data base



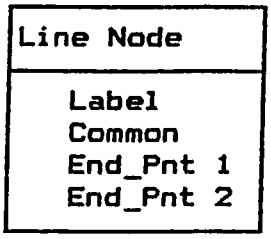
Information common to all entities

... Layer # of entity
Group # "
Color "
Density "
Symbol "
Entity type
Visibility switch



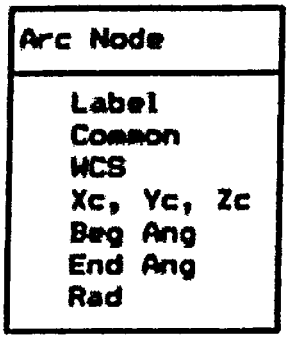
Defines a Point entity

... Point id
Share counter
Common information
Coordinates of point (ACS)



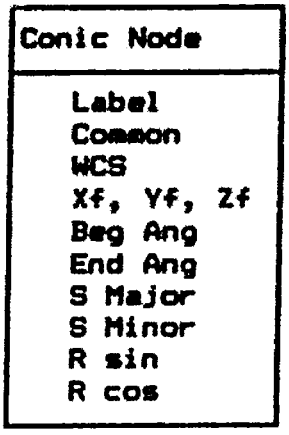
Defines a Line entity

... Line id
Common information
Pointer to end point
Pointer to end point



Defines an Arc entity

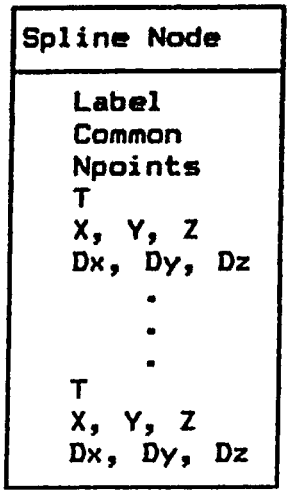
... Arc id
Common information
Pointer to local WCS
Center Coordinates (WCS)
Beginning angle
End Angle
Radius of arc



Defines a Conic entity

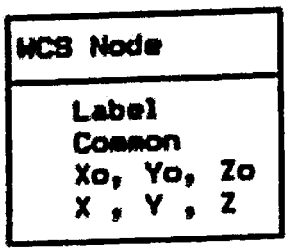
... Conic id
Common information
Pointer to local WCS

Parameter defenition depends on the type of conic being defined.



Defines a Spline entity
(Variable length node)
... Spline id
Common information
Number of points
Parameter value
Coord. of control point
Derivatives at point

Repeated for Npoints



Defines a Work Coordinate System

... WCS id
Common information

]- Define unit vectors

3.4.2 Storage Management:

The Pascal compiler used to develop Silouett, IBM Pascal 2.00, allocates a maximum of 64KB to the Constants, Heap (the dynamic memory area) and Data areas of memory. This means that the Heap area depends on the number of variables and constants defined within the program. Given the complexity of Silouett's code, the Heap area would be considerably less than 64KB. To avoid this restriction and to assure data base integrity, Silouett provides its own storage management environment in which:

1. Units of storage could be of different size and therefore storage requests must be accompanied by a size specification.

2. Individual data elements can be allocated and freed.

3. Storage no longer needed is returned to the system.

4. If storage is exhausted, requests for storage allocation are denied and an appropriate message is issued.

5. Free storage is organized as a link list of free storage blocks sorted by memory location (address).

6. When an entity is deleted, a block of storage is returned to the free list and an attempt is made to collapse adjacent free blocks into a single larger block.

7. Storage requests are granted using a first fit method of allocation.

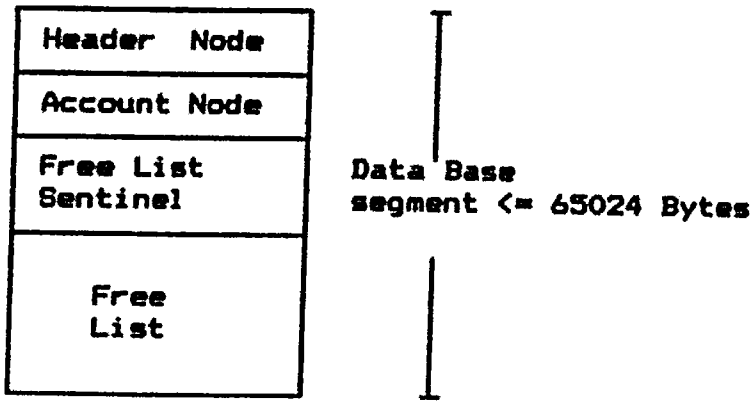
The data base management section includes procedures to:

1. Request memory from the Operating System - Requests from the operating system and reserves a segment of memory up to 64KB in

length. Multiple 64KB segments can exist within a data base file.

2. Initialize the data base -

Before any function can be performed, the data base must be initialized. This procedure calls upon the Request Memory procedure and organizes the returned memory segment as follows:



Note:

All available memory space after initialization is allocated to the free list.

3. Allocate memory blocks from free list -

Allocates a block of memory of specified length from the free list using the first fit method of allocation. The Free list is ordered by location, so the search for a

free memory block always starts at the Free list Sentinel.

4. Return memory blocks to free list -

The Memory Return procedure is the counterpart to the Memory Allocate procedure. When a block of memory is no longer needed, it must be returned to the free list so it can be re-allocated at a later time. By maintaining the free list sorted by memory location, it is possible to eliminate the need for a garbage collection algorithm. It is also possible to collapse adjacent free blocks into one continuous block. In this way, a good balance is obtained in memory even though storage is allocated and freed many times.

5. Save and load a data base to/from disk -

The Save and Load data base procedures are trivial procedures to transfer a data base back and forth between RAM memory and DISK. The only concern here are the link list pointers, but this is taken care of by the relocatable nature of the data base.

3.4.3 List Management:

The List Management section of the data base is responsible for maintaining the entities link list. There are six generic procedures to support this function:

1. Add_Entity

   The Add_Entity procedure adds a generic entity node to the corresponding entity link list. Added nodes are always inserted at the beginning of the link list.

2. Get_Entity

   Extracts information about a specified node in the data base. This routine is called by the applications program to traverse the data base.

3. Modify_Entity

   Allows a parameter of a given node to be modified. The application program should first call upon GET_Entity to get the current information about the node to be

changed and after modifying the entity data invoke Modify_Entity to restore the modified values.

4. Delete_Entity

The Delete_Entity procedure deletes a generic node from the corresponding entity link list, and returns the space occupied by the node to the free list.

5. Add_Entity_Label

The Add_Entity_Label procedure serves the same function of the Add_Entity procedure except that it allows the node to be identified by specifying its Label instead of its address.

6. Delete_Entity_Label

The Del_Entity_Label procedure serves the same function of the Del_Entity procedure except that it allows the node to be identified by specifying its Label instead of its address.

Additionally, one procedure for each entity supported is defined. The individual procedures customize the generic procedures before they are invoked.

Chapter 4 - Programming Considerations:

The major objective of this thesis work was to develop the foundations of a Three Dimensional Wire Frame CAD system for the IBM family of Personal Computers. A secondary objective was to develop the programs in such a way that they are easily transportable between other computer systems. To this end, whenever possible the approach was to use standard Pascal as the primary computer language.

Deviations from standard Pascal were taken only when there were no other reasonable alternatives. In such cases, the approach was to minimize the number of nonstandard Pascal procedures thus reducing the number of procedures that need to be changed when running the system in other than an IBM computer system.

The software is organized into separately compiled modules corresponding to logically separate functions of the system:

1. System Control Module
2. Planar Curves Module

3. **Display Control Module**
4. **Plotter Control Module**
5. **Global Storage Module**
6. **Data Base Module**

4.1 System Control Module:

To develop a commercially acceptable geometric modeling system within the time frame of a masters degree (18 months) is beyond feasibility for a single individual. For this reason the emphasis was put in developing the foundations of such a system so that it can be easily developed into a marketable product in the future. At this time, Silouett's system control module is really a test program used as a vehicle to demonstrate most of Silouett's current capabilities. The system architecture supports advance features that can not be demonstrated without a much more sophisticated control program. Some of this features are discussed in this section.

The main functions of the control program are:

- Manage the System Environment

- Input and Input Validation
- Command Menu Interaction
- Command Execution
- Data Base Traversal
- Output Device Control

p.The system process flow is diagramed in Figure 12 on page 56.

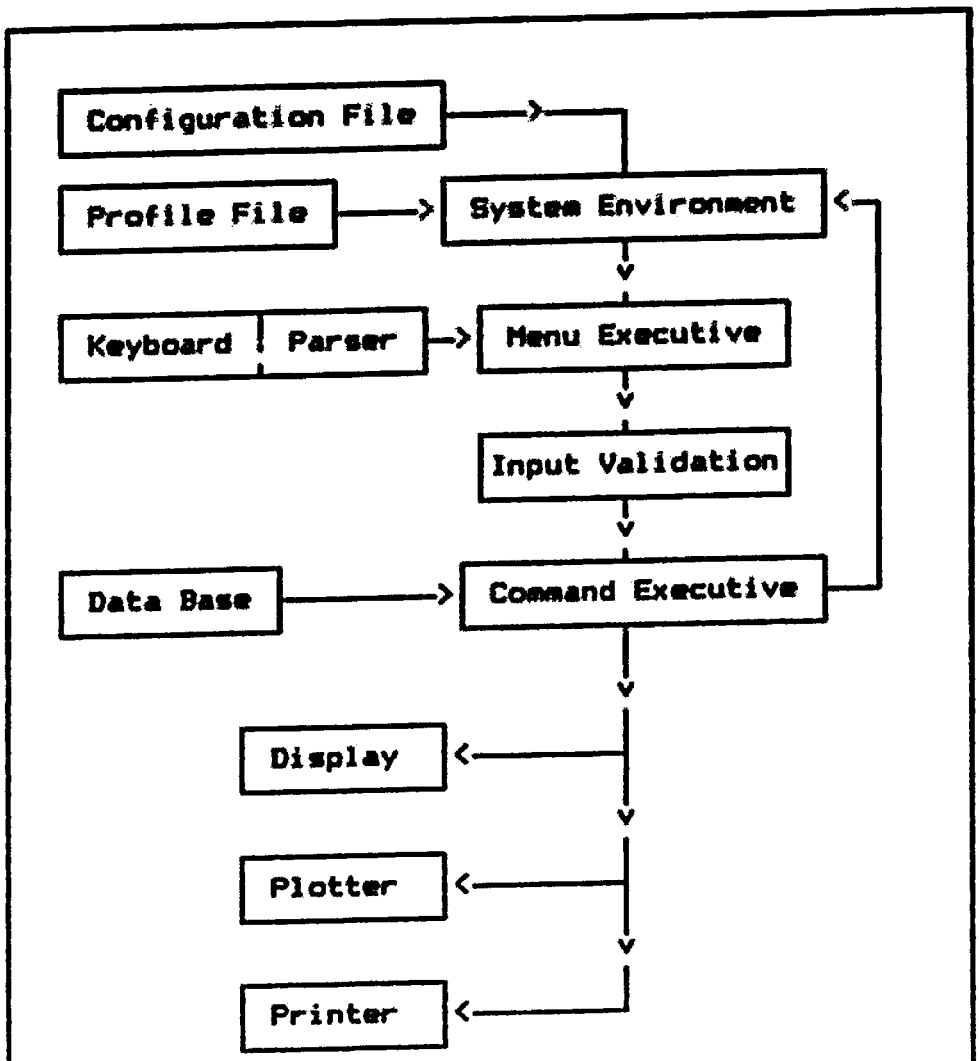


Figure 12. System Process Flow

4.1.1 System Environment Management:

The user controls the system environment through three different methods:

1. Configuration File

   The configuration file is executed when the system is first boot up. The purpose of this file is to tell the operating system which device drivers are to be loaded, which ones can reside in an overlay segment and which ones must be permanently loaded. It is also used to further customize the operating system environment.

2. Environment Variables

   Silouett supports a series of environment variables such a:

   - Auto scale
   - Auto erase
   - Display borders
   - Reference coordinate system
   - System units

- Graphic entities attributes

The user can toggle or specify a value for most of the control variables at any time during the work session. Some variables, like system units, can only be specified when the model is first created.

3. Profile File

All of the control variables can be assigned default values in a file know as the Profile file.

4.1.2 Input and Input Validation:

All interaction with the user is accepted by the system in alphanumeric form. This approach eliminates the possibility of a system crash due to variable format mismatch. The input is then processed by a command parser which breaks the input line into an alphanumeric command and its corresponding numeric parameters.

The command parser allow the user to default

intermediate parameters (the system defaults them to zero) but insist on a last parameter if any are defaulted.

Once the input line is processed, the parser passes the command and parameters to the menu executive procedure.

4.1.3 Command Menu Interaction:

Command menus are organized according to the main function being executed at the time. Function menus are divided into a global menu and several local or functional menus. The global menu processes commands that must be active at any time during a session, while local menus process commands related to a local function.

When the menu executive procedure receives a command from the command parser, it first tries to execute in through the local command executive. If the command fails, the system tries to execute it as a global command. If a failure still occurs, a message is issued to the user indicating that the command is unknown to the system. Since at any one time the

system can only recognize a set of local commands and the global commands, attempting to execute a local command from a nonactive local command menu results in a command unknown message.

4.1.4 Command Execution:

Once the menu executive procedure determines that a valid command has been requested, it calls on the command executive to execute the command. At this time a second input validation takes place, this time to ensure that the parameters passed are within the acceptable range of the requested function. If they are not, the system informs the user and requests parameter input again.

After command and parameters have been checked, the command executive procedure executes the requested command.

4.1.5 Data Base Traversal:

Silouett's architecture supports the concept of a Draw File but time did not allowed its implementation.

A Draw File is a means to greatly speed up the time it takes to redraw the model on the screen. If the data base has to be traversed every time the image of the model is redrawed on the output device, the processing time would probably be unacceptable. The viewing process previously described showed the series of transformations and mappings that take place when the data base is traversed. We also have previously indicated that eventually the model is broken down into a series of line segments and or points. The Draw File contains these line segments and points that describe the current image of the model. Therefore the Draw File can be redrawed without going through the viewing process, cutting out most of the process overhead. The Draw File must be regenerated after certain model transformations are requested.

Currently Silouett traverses the database every time the image of the model is redrawed.

4.1.6 Output Device Control:

The final function of the control program is to

direct the output of graphic primitives to the output device requested by the user.

4.2 Planar Curves Module:

All planar curves supported by Silouett are conic sections.

As mention before, all graphic entities can be broken down into a series of point and or line segments. Therefore, whenever the control program detects that a conic section is to be displayed, the planar curves module is called to supervise the operation.

The responsibility of planar curves module is to break down the conic section into a series of line segments for the display module to process.

The logic followed by Silouett's planar curve module was first described in [OZSO83], however the actual implementation is different. To display a planar curve, the planar curves module has to:

1. Define the plane of the curve.
2. Define the curve in parametric form.

direct the output of graphic primitives to the output device requested by the user.

4.2 Planar Curves Module:

All planar curves supported by Silouett are conic sections.

As mention before, all graphic entities can be broken down into a series of point and or line segments. Therefore, whenever the control program detects that a conic section is to be displayed, the planar curves module is called to supervise the operation.

The responsibility of planar curves module is to break down the conic section into a series of line segments for the display module to process.

The logic followed by Silouett's planar curve module was first described in [OZS083], however the actual implementation is different. To display a planar curve, the planar curves module has to:

1. Define the plane of the curve.
2. Define the curve in parametric form.

3. Determine the number of line segments required to properly define the curve.
4. Display the curve by calling the appropriate procedures from the display module.

4.2.1 Defining a Working Plane:

Planar curves by definition are two dimensional curves. However, we would like to be able to define the plane of the curve anywhere in space, we call this plane a working plane.

To display the planar curve, Silouett makes use of the concept of projection matrices. In our case, a projection matrix is defined as a matrix that projects a vector in space onto a plane defined by the user. The reverse operation can be obtained by transposing the projection matrix.

By convention, all planar curves created with Silouett, are created in the X-Y plane of a local work coordinate system (WCS). Therefore the first step in creating a curve is to define the reference WCS to be used.

To define a WCS, the user must specify, in some way, the unit directional cosines of two of the tree WCS axis. This is usually done by specifying the ACS coordinates of: the WCS origin, a point in its X axis, and a point in either the X-Y or X-Z planes. From this information the system computes the unit directional cosines of the WCS x and Y axis, thus defining the X-Y plane of the curve and the required projection matrix.

The projection matrix, P, has the following form:

$$P = \begin{bmatrix} X\alpha & X\beta & X\mu \\ Y\alpha & Y\beta & Y\mu \end{bmatrix}$$

Where: [Xα, Xβ, Xμ] define the unit directional cosines in the X direction.

[Yα, Yβ, Yμ] define the unit directional cosines in the Y direction.

4.2.2 Parametric Representation of Curves:

The nonparametric representation of cures either can not be used to represent closed or multiple value curves, or require a root finding algorithm to determine the coordinates of a point on the curve. Also nonparametric equations are axis dependent. For

these reasons, we use a parametric representation of curves in geometric modeling.

In parametric representation, each coordinate of a point is represented as a function of one or more parameters. The positional vector of point on the curve is thus determined by the value of the parameters. This makes parametric curves suitable for representing closed curves and curves with multiple values at a given value of the independent variable. Since a point on the parametric curve is specified by the value of its parameters, the parametric form is axis independent.

For a more rigorous discussion on parametric equations, the reader is referred to any advance calculus book. A more pertinent discussion of the topic can be found in [ROGE76].

4.2.3 Number of Points to Define a Curve:

The selection of the number of points to define a curve is affected by:

1. The parametric representation of the curve.

2. Desired efficiency of computations.

3. Desired point distribution on the curve.

All conic sections can be easily represented in parametric form in terms of the parameter θ, that is, the angle sustained by a vector from the conic focus to the point on the curve.

This parametrization allows us to make use of the trigonometric double angle formulas to speed up the computations.

Recall the double angle formulas are:

$$\cos(\theta + d\theta) = \cos(\theta)\cos(d\theta) - \sin(\theta)\sin(d\theta)$$

$$\sin(\theta + d\theta) = \cos(\theta)\sin(d\theta) + \cos(d\theta)\sin(\theta)$$

Therefore, if we fix the incremental angle $d\theta$ between different values of the parameter θ, we only have to compute the trigonometric functions once.

For example, in the case of a circle centered at the origin, once the selection of the incremental angle

$d\theta$ is made, we can define the parametric equations as follows:

For the initial point,

$$Xn = R \cos(\theta)$$
$$Yn = R \sin(\theta)$$

For all other points,

$$Xn+1 = Xn \cos(d\theta) - Yn \sin(d\theta)$$
$$Yn+1 = Xn \sin(d\theta) + Yn \cos(d\theta)$$

Again, since $d\theta$ is constant, so are $\cos(d\theta)$ and $\sin(d\theta)$ and therefore are only computed once.

The only remaining factor is the point distribution. A fixed number of points can be use to describe the incremental parameter $d\theta$, however this could result in either over or under specified curves. The converse solution, specifying $d\theta$, and then computing the number of points suffers from identical problems. To address this issue, we use the critical radius (R) of the curve to determine the number of points (N) to be used. Then, from the number of

points and the curve sweep angle (Σ) the incremental parameter $d\theta$ is computed.

We still need some sort of criteria to determine the number of points from the critical radius. The criteria used by Silouett, is to control the perpendicular distance (t) between the the bisector to a chord at the critical radius and the curve. This method is superior to specifying the chord angle, as it preserves the smoothness at all scales.

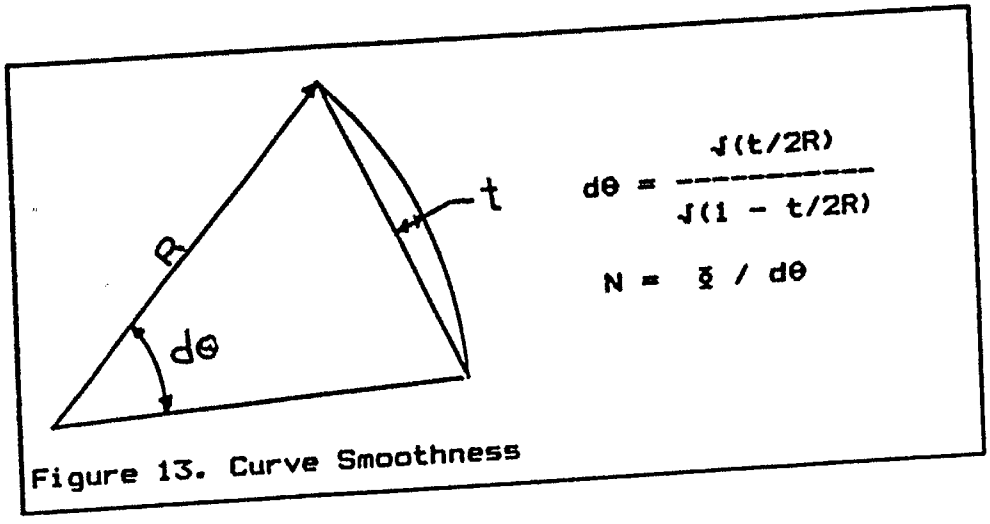


Figure 13. Curve Smoothness

The following is a brief description of the main procedures in the planar curves module.

4.2.4 WCS_XY and WCS_XZ:

Defines a new work coordinate system and add it to the data base for future references.

Description:

Given the ACS coordinates of the WCS origin, a point in the local X axis and a point on the XY (XZ) plane. The procedure uses vector cross products to compute the directional cosines required to define the WCS and the projection matrix P previously described.

4.2.5 Transform_WCS_ACS:

Transforms the WCS coordinates of a point to its corresponding ACS coordinates and viseversa. To allow more flexibility in its use, the procedure assumes both coordinate system have the same origin, if this is not the case, the user can apply a translation to the transformed ACS of the point to get the true ACS coordinates.

Description:

This procedure simply apply the projection matrix or its transform to the given input.

4.2.6 Circle:

Draws a circle at the specified location and of a given radius on a plane specified by user. WCS_XY or WCS_XZ must be invoked to define the working plane and obtain a WCS_ID before invoking this procedure.

Description:

Uses a parametric representation of the circle to break it up into a number of equal angle segments. The number of segments is proportional to the radius, scale of model and system of units.

Parametric equations of circle :

$$Xn = R \cos(\theta)$$

$$Yn = R \sin(\theta)$$

After aplying double angle formulas,

$$Xn+1 = Xn \cos(d\theta) - Yn \sin(d\theta)$$

$$Yn+1 = Xn \sin(d\theta) + Yn \cos(d\theta)$$

4.2.7 Arc_Angle:

Draws an Arc at the specified location and of a given radius, beginning, and end angles on a plane specified by user. WCS_XY or WCS_XZ must be invoked to define a working plane and obtain a WCS_ID before invoking this procedure.

Description:

Uses a parametric representation of the Arc to break it up into a number of equal angle segments. The number of segments is proportional to the radius, scale of model and system of units.

Parametric equations of arc :

$$Xn = R \cos(\theta)$$

$$Yn = R \sin(\theta)$$

After aplying double angle formulas,

$$Xn+1 = Xn \cos(d\theta) - Yn \sin(d\theta)$$

$$Yn+1 = Xn \sin(d\theta) + Yn \cos(d\theta)$$

4.2.8 Arc_Edge:

Given the coordinates of the center and a point on the limiting edge of the arc, this procedure draws an Arc from the current beam position to the Edge defining the end of the Arc. This Edge is defined by the center of the Arc and a point on the Edge. The radius is the length of the line from the center to the last beam position. WCS_XY or WCS_XZ must be invoked to to obtain a WCS_ID and define a working plane with X axis aligned with the arc center and current beam position.

Description:

Uses a parametric representation of the Arc to break it up into a number of equal angle segments. The number of segments is proportional to the radius, scale of model and system of units.

Parametric equations of arc :

$$Xn = R \cos(\theta)$$

$$Yn = R \sin(\theta)$$

After aplying double angle formulas,

$$Xn+1 = Xn \cos(d\theta) - Yn \sin(d\theta)$$

$$Yn+1 = Xn \sin(d\theta) + Yn \cos(d\theta)$$

The procedure first determines the angle between the vectors defined by : Current beam position - Center Point and Edge - Center, it then divides this angle into segments and proceeds to draw them.

4.2.9 Arc_Three_Points:

Given the ACS coordinates of three points in space; the beginning point, a second point on the arc and the end point, this procedure will draw an Arc through them.

Description:

The center of the Arc and the sweep angle are computed from the coordinates of the three given points, then Arc_Angle is invoked to draw the arc.

Parametric equations of arc :

$$Xn = R \cos(\theta)$$

$$Yn = R \sin(\theta)$$

After aplying double angle formulas,

$$Xn+1 = Xn \cos(d\theta) - Yn \sin(d\theta)$$

$$Yn+1 = Xn \sin(d\theta) + Yn \cos(d\theta)$$

4.2.10 Parabola:

Draws a parabolic section given the coordinates of the vertex, the maximum and minimum values and the rotational angle (Ω) in the plane of the parabola.

Description:

The distance along the directrix between the parabola maximum and minimum value is divided into segments of equal length. The end points of this segments are

then projected onto the parabola to obtain the coordinates of the points on the parabola between which line segments are drawed.

Parametric equations:

$$Xn = a\theta$$

$$Yn = 2a\theta$$

a = focal distance

After applying in plane rotation,

$$Xn+1 = Xn \cos(\Omega) - Yn \sin(\Omega)$$

$$Yn+1 = Xn \sin(\Omega) + Yn \cos(\Omega)$$

4.2.11 Ellipse:

Draws a section of a Ellipse specified by the beginning and end angles, semi_major and semi_minor axis lengths, and inplane rotational angle (Ω).

Description:

Uses a parametric representation of the Conic to

break it up into a number of equal angle segments. The number of segments is proportional to the radius, scale of model and system of units.

Parametrization :

$$Xn = K1 \cos \theta$$

$$Yn = K2 \sin \theta$$

K1 = Length of semi_major axis

K2 = Length of semi_minor axis

After adjusting for rotation,

$$Xr := Xn \cos(\Omega) - Yn \sin(\Omega)$$

$$Yr := Xn \sin(\Omega) + Yn \cos(\Omega)$$

After applying double angle formulas,

$$Xn+1 = Xr \cos(d\theta) - Yr * (K1/K2) \sin(d\theta)$$

$$Yn+1 = Xr * (K2/K1) \sin(d\theta) + Yr \cos(d\theta)$$

4.2.12 Hyperbola:

Draws a section of a Hyperbola specified by the beginning and end angles , semi_transverse and

semi_conjugate axis lengths, and inplane rotational angle (Ω).

Description:

Uses a parametric representation of the Conic to break it up into a number of equal angle segments. The number of segments is proportional to the radius, scale of model and system of units.

Parametrization :

$$Xn = K1 / \cos\theta$$

$$Yn = K2 \sin\theta / \cos\theta$$

K1 ≡ Semi_transverse axis length

K2 ≡ Semi_conjugate axis length

After adjusting for rotation,

$$Xr := Xn \cos(\Omega) - Yn \sin(\Omega)$$

$$Yr := Xn \sin(\Omega) + Yn \cos(\Omega)$$

After applying double angle formulas,

$$Xn+1 = Xr * K2 / [K2 \cos(d\theta) - Yr \sin(d\theta)]$$

$$Y_{n+1} = K2 * [Yr + K2 \tan(d\theta)]/[K2 - Yr \tan(d\theta)]$$

4.3 Display Control Module:

The display module is responsible for producing the image of the object model on the output device. The procedures required to accomplish this task can be divided into three functional groups:

1. Attribute Setting

   The procedures within this group allow the user to set and modify graphic entity attributes such as color, layer number, group number, etc.

2. Drawing Functions

   All graphic entities can be broken down into a series of points and or line segments. Therefore the only procedures that need to communicate with output device are the draw point and draw line procedures. More complicated shapes, such as the planar curves, to be described later, interface with these procedures to generate there images.

3. GTM Functions

The procedures in this group operate on a General Transformation Matrix, GTM, to change the image of the object on the logical view surface.

The following is a brief description of the main procedures in the display module.

4.3.1 Display Initialize:

This procedure initializes the display driver and defines view volumes and viewports.

Description:

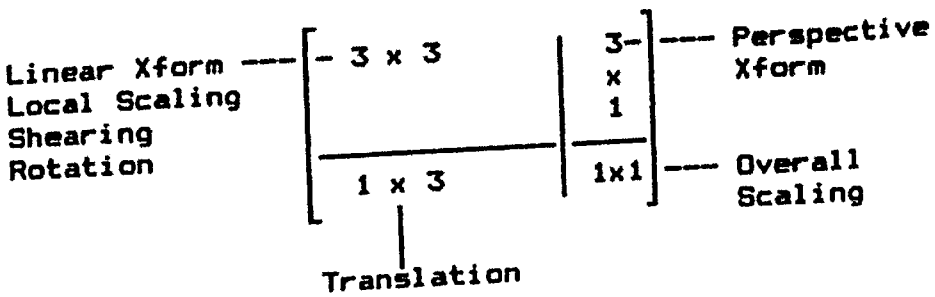
When the display driver is initialized, the physical size of the screen is determined and the limits of the GCS set accordingly. Once this is accomplished, the display viewing surface is formatted into logical view surfaces. Finally view volumes and viewports are defined and default views assigned to them.

4.3.2 Transform:

Transforms the coordinates of a point from the Absolute Coordinate System (ACS) to the Global Coordinate System (GCS).

Description:

Applies a 4 x 4 GTM to the ACS coordinates of a point to yield the normalized transformed coordinates in the GCS. This transformation produces a combination of shearing, local scaling, rotation, reflection, translation, perspective and overall scaling.



4.3.3 Mapping:

Maps points from Global Coordinate System to Normalized Device Coordinates per specified Viewport parameters.

Description:

Before any entity can be draw, it must be mapped to the output device. This requires a transformation from GCS to NDC. Since the Viewport is geometrically similar to the Global Coordinate Space and further more their origins are shifted in the same proportion, the GCS to NDC transformation is simplified to:

NDC = GCS / SF where : SF = Delta GCS / Delta NDC

The viewport definition contains the required scale factor, SF, required to accomplish this simple transformation.

4.3.4 Clipper:

Redefines the endpoint coordinates of a line intersecting the specified view volume such that the line lies within the view volume limits. Clipping is performed in global coordinates.

Description:

The procedure uses the Cohen-Sutherland clipping algorithm. The algorithm is designed to identify efficiently those lines that can be trivially accepted or rejected by using region checks. If the line can not be accepted or rejected, the algorithm uses a divide and conquer strategy to home in on the intersection point.

Algorithm:

For a detail description of the algorithm refer to [FOLE82].

4.3.5 Move_Absolute:

Updates the current position of the graphics cursor.

Description:

The current position of the graphics cursor is maintained in a global variable named CP. This variable is updated every time the graphics cursor is moved. The ACS and GCS coordinates of CP are updated but not the NDC since this will change if clipping is required. Clipping is different for Points and

Lines, therefore the NDC coordinates of CP are updated by the Point_Absolute and Line_ Absolute procedures.

4.3.6 Point_Absolute:

Draws a point on the output device at the absolute location specified. Point is addressed in ACS.

Description:

This procedure will draw a point on the current output device or erase an existing point on the screen. If the user can guarantee before hand that the point lies within the view volume boundaries (ie... animation applications, etc) then the call is made avoiding the clip check.

Process Flow:

1. Move graphics cursor to specified point.
2. If Clip check was requested, then check if point lies within the current view volume. If it lies outside, exit procedure now, else continue.
3. Map point GCS coordinates to NDC.

4. Draw point.

4.3.7 Point Relative:

Draws a point on the output device relative to the last graphics cursor position. Point is addressed in ACS.

Description:

This procedure will draw a point on the current output device or erase an existing point on the screen. If the user can guarantee before hand that the point lies within the view volume boundaries (ie... animation applications, etc) then the call is made avoiding the clip check.

Process Flow:

1. Compute point coordinates by adding the specified relative coordinates to the current cursor position.
2. Invoke Point_Absolute

4.3.8 Line_Absolute:

Draws a line on the output device from the current graphics cursor position to the point specified in ACS. Before this procedure is called, the system must ensure that no view volume changes have occurred since the last time an entity was draw. One way to guarantee this is to execute the Move_Absolute procedure before Line_Absolute.

Description:

This procedure will draw a line on the current output device or erase an existing line on the screen. If the user can guarantee before hand that the line lies within the view volume boundaries (ie... animation applications, etc) then the call is made avoiding the clip check.

Process Flow:

1. Pick up graphics cursor current GCS coordinates.
2. Apply the GTM corresponding to the current view to the line end points.

3. If user specified so, apply clipping process to end point coordinates.
4. Map end point GCS coordinates to NDC.
5. Draw line segment.

4.3.9 Line_Relative:

Draws a line on the output device a relative distance away from the current graphics cursor position, CP. The relative position of the end point is addressed in ACS. This procedure assumes no view volume changes have occurred since the last call to a procedure that updates CP.

Description:

This procedure will draw a line on the current output device or erase an existing line on the screen. If the user can guarantee before hand that the line lies within the view volume boundaries (ie... animation applications, etc) then the call is made avoiding the clip check.

Process Flow:

1. Compute line end point coordinates by adding the specified relative coordinates to the current cursor position.
2. Invoke Line_Absolute to draw the line segment.

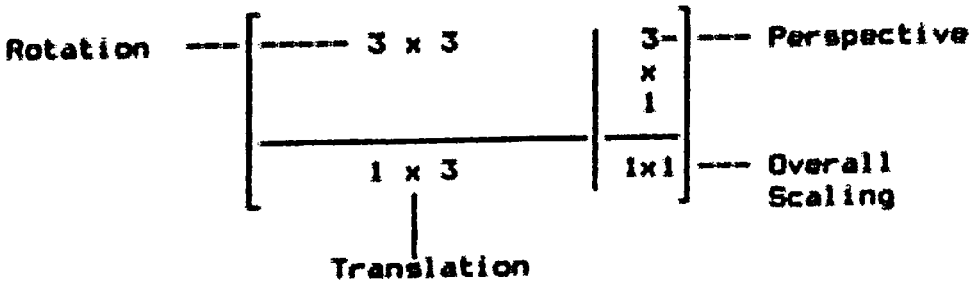
4.3.10 Set_GTM:

Sets the general transformation matrix for the specified View.

Description:

Allows the user to specify an arbitrary transformation matrix and stores it in the specified View array.

The parameters passed are used to define a 4 x 4 transformation matrix as follows:



4.3.11 Set_GTM_Relative:

Performs matrix multiplications for model transformations.

Description:

This routine Pre or Post multiplies a view GTM by a second transformation matrix, and replaces the contents of the view GTM with the results. The order of multiplication determines if the transformation takes place with respect to the ACS or the GCS.

4.3.12 Scale_GTM_Relative:

Enables the user to either enlarge or shrink the picture by specifying an overall scale factor.

Description:

This procedure simply changes the Scale factor of the corresponding transformation matrix. The scaling process is relative to the current scale factor.

Process Flow:

1. Create a temporary transformation matrix with a scale factor as specified by the user.

2. Invoke Set_GTM_Relative to actually change the scale factor of the corresponding view.

4.3.13 Translate_GTM_Relative:

Enables the user to translate the object image relative to either GCS or ACS.

Description:

Redefines the transformation matrix corresponding to the specified VIEW.

Process Flow:

1. Create a temporary transformation matrix with the translation factors as specified by the user.

2. Invoke Set_GTM_Relative to translate the image in the corresponding view.

4.3.14 Rotate_GTM_Relative:

Enables the user to rotate the object image relative to either GCS or ACS.

Description:

Redefines the transformation matrix corresponding to the specified VIEW.

Process Flow:

1. Create a temporary transformation matrix with the rotational factors as specified by the user.

2. Invoke Set_GTM_Relative to rotate the image in

the corresponding view.

4.3.15 Copy_GTM:

Copies the general transformation matrix defining one view into the general transformation matrix defining a second view.

4.4 Plotter Control Module:

The plotter control module acts as an executive command supervisor to the display module during plotting operations. It controls the plotting device and format of the plots.

The user can select two basic modes of operation:

1. Display Dump Mode - The display screen is dumped to the plotter. During this operation if the physical size of the plotter is different to the physical size of the screen, a scaling operation is performed. This scaling maintains aspect ratios but not the actual scale of the

drawing. This operation requires no interaction from the user as it is intended to produce a quick plot.

2. Detail Plot Mode - Under this mode of operation the user has much more control over the final plot. The procedures discussed in this section apply to this mode of operation unless otherwise so stated.

When the user first invokes the plotter control module, the current screen display is temporarily replaced by an outline of the plotting surface. At this point the user interacts with the system to format the plot. Once the plotting process is completed, the system returns to the screen format active just before the plotter control module was invoked.

The following is a brief description of the interactive plot procedures available to the user to format the plot.

1. Select View

Allows the user to bring into the screen a view to be plotted.

2. Reject View

Allows the user to reject or cancel a previously selected view.

3. Relocate View

Allows the user to relocate a view anywhere within the outline of the plotting surface.

4. Plot Views

Plots all currently selected views. The views are positioned in the plotting surface exactly as indicated in the plotting surface outline in the screen.

At any one time the user can change or specify the scale of the drawing.

4.5 Global Storage Module:

All software modules have access to two data segments at any time:

1. Data Base
2. Global Storage

While the data base maintains all the information relevant to the geometric model, it only maintains essential information about the geometric modeling system environment. This information is dynamically build and maintained by the application programs. The global storage data segment is the area of memory designated to contain this information.

The global storage area can be subdivided as follows:

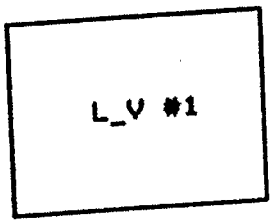
1. Viewing Volumes and Viewports
2. View Data
3. Cursor Current Position
4. Check Volume
5. Peripheral Information

6. Global Data

4.5.1 Viewing Volumes and Viewports:

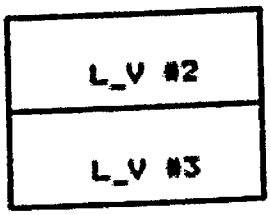
Silouett allows the user to format the viewing surface of the output device in four different ways:

1. Single View - The entire view surface is used to display a single logical view. Any model view can be projected onto the logical view surface.



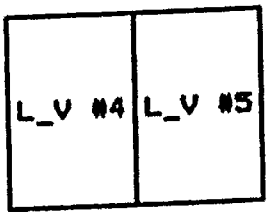
View Surface

2. Two Views, Horizontal Split - The view surface is split in two logical viewing surfaces. A different model view can be projected onto each logical view surface.



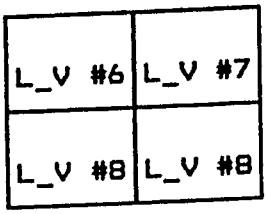
View Surface

3. Two Views, Vertical Split - The view surface is split in two logical viewing surfaces. A different model view can be projected onto each logical view surface.



View Surface

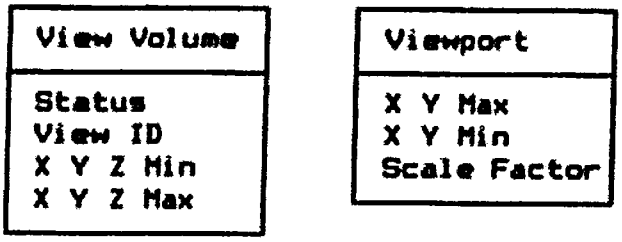
4. Four View Split - The view surface is split in four ways. A different model view can be projected onto each logical view surface.



View Surface

For each logical view, a view volume and viewport must be defined. Further more, the system must have a

means of finding out which particular format is active at any one time and which view is being projected on it. To this end, Silouett defines the view volumes and viewports as follows:



Where:

Status	Indicates if the view volume is currently active or not.
View ID	Identifies the view to be projected onto the viewport corresponding to the particular view volume. There is a one to one correspondence between view volumes and viewports.
Scale Factor	The scale factor is used to map global coordinates to normalized device coordinates.

The other fields define the view volume and viewport to the system as described in the Viewing in Three

Dimensions chapter.

4.5.2 View Data:

In order to obtain different views of the model, we simply apply a predefined general transformation matrix (GTM) to the data base before we display it on the output device. Therefore if we want to support multiple views, all we have to do is provide a storage area to hold the definition of the GTM that defines the view. Silouett allows the user to define up to 64 different views and internally defines nine additional views called the standard views. These standard views allow the user to reset any view to a predefined format, ie... isometric, front view etc. The views are identified by view number.

4.5.3 Current Cursor Position:

The current graphics cursor position (CP) is maintained in three coordinate system, ACS, GCS and NDC. The current cursor position is maintained to support relative draw procedures. These procedures will be discussed later in this chapter.

4.5.4 Check Volume:

Silouett supports a function called Autoscaling. The purpose of this function is to apply a view scaling factor such that the model is displayed on the center of the logical view surface with maximum dimensions. For this purpose we must keep track of the maximum coordinates of all entities that constitute the model. This information is kept in the global storage area.

4.5.5 Peripheral Information:

The system keeps track of all peripheral devices attached to the system at any one time. Vital information such as physical size, addressability and graphics capabilities are recorded for future reference.

4.5.6 Global Data:

Finally, the last area of global storage is reserved to keep miscellaneous information required by the control program.

4.6 Data Base Module:

As was indicated in the Data Base chapter, Silouett implements its own storage management environment. However, from an applications programming point of view, the user does not have to concern itself with how the data is maintained. The database is interfaced through the List Management functions which allow the user to add, get, modify, and delete entities from the database.

CONCLUSIONS

Computer Aided Design and Manufacturing, can be described as the effective utilization of computers in the process of designing and manufacturing a product or system. The rapid growth of the CAD/CAM industry can be attributed to the increased productivity and reduction of production cost that can be realized by a well managed CAD/CAM facility. In computer aided design (CAD), interactive graphics are used to design components and systems of mechanical, electrical, and structural nature. Initially the emphasis was solely on the automated drafting aspects provided by CAD, but more recently the emphasis has been on interacting with the computer-based model of the component or system being designed in order to analyze and test the design before a single piece of hardware is produced. The final evolution of CAD/CAM, known as Computer Integrated Manufacturing (CIM) is to provide computer assistance in all business functions from initial marketing research to product shipment. It embraces among other things, business planning and support, engineering design and analysis, manufacturing process planning, manufacturing process control, shop

floor monitoring systems, and process automation.

The post-processing capabilities of CAD packages, depend to a great extend on the method used to represent the geometry of the model. Of the three basic approaches to geometric modelling: wireframe, surface, and solid modelling, wireframe provides the least amount of information about the object being modeled, but because it is also the least computationally intensive of all, remains the most stable, enhanced and familiar means of computer aided design.

Although some vendors claim to have develop solids modeling systems for personal computers, it is the believe of the author that personal computers, without the help from dedicated hardware processors that effectively convert then into mini computers, can not handle yet the computational load impose by such modeling systems.

In the development of Silouett, I have tried to keep in mind the possibility of expanding the system into a surface modeling system in the future. I believe this would be a more reasonable expectation for

personal computers.

As far as mainframe modeling systems are concern, it is evident that solids modeling will be the common practice rather than state of the art in the near future.

REFERENCES

[ATRE80] Data Base : Structured Techniques for Design, Performance, and Management
S. Atre
John Wiley & Sons, Inc.
(c) 1980

[CODD70] A relational Model of Data for Large Shared Data Banks
E. F. Codd
Communications of the ACM, Vol 13
(c) June 6, 1970

[ELSO75] Data Structures
Mark Elson
Science Research Associates
(c) 1975

[FOLE82] Fundamentals of Interactive Computer Graphics
James D. Foley
Andries Van Dam
Addison Wesley Publishing Co.
(c) 1982

[GIL078] Interactive Computer Graphics
Wolfgang K. Giloi
Prentice Hall, Inc. NY
(c) 1978

[KNUT68] The Art of Computer Programming
Volume 1 : Fundamental Algorithms
Donald E. Knuth
Addison Wesley Publishing Co.
(c) 1968

[OZS083] VS11-3D Graphics Package
Reference Manual
Examples Manual
Tulga M. Ozsoy
Sunil Bhalla
Randi Summer
Lehigh University, 1983

[OZS084] Data Base Manipulation Routines
Reference Manual
Tulga M. Ozsoy
Leyla Kivanc
Lehigh University, 1984

[ROBE65] Homogeneous Representations and Manipulations of N Dimensional Constructs

Roberts, L. G.

Document MS 1405

Lincoln Laboratory,

Massachusetts, 1965

[ROGE76] Mathematical Elements for Computer Graphics

David F. Rogers

J. Alan Adams

McGraw-Hill Book Co.

(c) 1976

[SUTH63] SKETCHPAD: A Man-machine graphical communication system.

Sutherland, I. E.

SJCC 1963,Spartan Books

Baltimore, Md

APPENDIX A - SUMMARY OF SYSTEM COMMANDS

Functional Description:

Silouett is a three dimensional geometric modeling system for personal computers, that allows the user to interactively build a 3D Wire Frame model of an object.

To define the model, the user can make use of the following graphic primitives:

- **Points**
- **Lines**
- **Arcs**
- **Circles**
- **Parabolas**
- **Ellipses**
- **Hyperbolas**

There are two coordinate system the user should be aware off:

1. Global Coordinate System (GCS)

   The GCS is attached to the lower left corner of

the screen and uses millimeters to define coordinates.

2. Absolute Coordinate System (ACS)

Is a right handed 3D coordinate system attached to the model. The system of units of the ACS is specified by the user in the Profile File by specifying a calibration factor corresponding to the number of millimeters per user unit.

System Configuration:

The following is a listing of the minimum system configuration requirements to run Silouett. The equipment labeled "Optional" is optional but recommended for maximum performance.

Processor : IBM Personal Computer (PC,Jr,XT,AT)
256 KB Dynamic Ram (Minimum)
360 KB Diskette Drive
10 MB Disk Drive (Optional)
Color Graphics Adapter
Enhanced Graphics Adapter (Optional)
Asynchronous Communications Adapter
8087 Math Co-Processor (Optional)

Displays : IBM Enhanced Color Display (Optional)
IBM Color Display
IBM Monochrome Display (Optional + EGA)

Peripherals: IBM Graphics Printer
IBM 7372 6 Pen Plotter
IBM Color Graphics Printer (Optional)

Profile File:

The profile file is a diskette resident file that the user can edit in order to customize Silouett's environment.

```
**********************************************************
*                SILOUETT's  PROFILE                     *
* ------------------------------------------------------ *
* Do not alter the sequence of the profile,processor     *
* is line and colum number sensitive.                    *
**********************************************************
Value Parameter                Remarks
-----  ----------------------  ------------------------------
0      Background Color        See Users Manual: Attribute  *
1      Point      Color                          "          *
3      Line       Color                          "          *
6      Text       Color                          "          *
1      Point      Symbol                         "          *
1      Line       Style                          "          *
1      Text       Style,                                    *
1      Line       Width       Plotter Only                  *
1.0    mm/User units          Conversion Factor             *
1.0    mm                     Curve Smoothness factor       *
**********************************************************
*                End of Profile                          *
**********************************************************
```

Users Interface:

At the present time, keyboard input is the only form the user has to command Silouett. At any one time, a menu of the currently active commands is displayed on the screen for the users reference. Once the user selects a command, the system will prompt him for the required input. As will be described latter, global commands can be executed at any time.

Only the first three letters of the displayed command name need to be specified, if more are given, Silouett will ignore them.

Parameters can be defaulted to zero by replacing them with a comma.

4.1 Global Commands:

Global commands are the most frequently invoked commands. When the system is boot-up, the global commands menu is displayed for the users reference. Some of the global commands require the menu to be changed, when this happens, the global commands menu

will be replaced by a local commands menu. However, global commands are still recognized by Silouett.

The following are the global commands supported by Silouett, only the capitalized portion of the command name needs to be specified to execute the command.

4.1.1 AUTo scale:

Scales the model so that it fits within 90% of the maximum display area of the current window.

4.1.2 REDraw:

Redraws the image of the model in the current window.

4.1.3 ROTate Rx Ry Rz:

Rotates the image of the model, relative to the active coordinate system. If the rotational angles are not given, the system will prompt the user again. The rotational angles are specified in degrees.

1) ROT 10 20 30 or

ROT 10,20,30

Rotates the model 10° about the X axis, then
20° about the Y axis, then
30° about the Z axis.

2) ROT ,,30

Rotates the model 30° about the Z axis.

4.1.4 SCAle SF:

Scales the image of the model relative to the active coordinate system. If the scale factor, SF, is not specified the system will prompt the user again.

The user can fix the scale of the model by not specifying the SF, when the system prompt again for the SF, prefix the SF with the prefix FIX.

1) SCAle 2.5

The new scale of the model will be 2.5 times

the current scale.

2) Scale

Enter new scale factor ==> Fix 1

The scale of the model is changed to full scale.

4.1.5 STOre VN:

Allows the user to store the current view into a new reference view. If the reference view number VN, is not given or is not valid, the system will prompt the user again.

1) STOre 20

Stores the current view in reference view number 20. The current view is not changed.

4.1.6 SWItches:

Allows the user to change environmental parameters. The current active menu is temporarily replaced by the SWITCH menu.

The supported switches are:

ON	Turns all switches on.
OFF	Turns all switches off.
ABCale	Toggles the auto scale switch. When on, the current view will be automatically auto scaled whenever the image is redrawed or any transformation is applied to the model. Off disables the automatic auto scale call.
AUTo erase	Toggles the autoerase switch. When on, the current view is erase whenever a transformation is applied to the model. When off, causes the model image to be superimposed on the previous image.
BORders	When on, the system will draw an outline of the active windows. When off, no outline is drawed.
COOrdinate	Allows the user to select the active

coordinate system.

When on, the ACS is active, when off, the GCS is active. Scale, Rotate and Translate are affected by this switch.

4.1.7 TRAnslate Tx Ty Tz:

Applies a translation factor to the image of the model. If the translational factors are not specified, the system will prompt the user again.

1) TRAnslate 10 10 0

Translate the image of the model,
10 units in the X axis direction, and
10 units in the Y axis direction, and
0 units in the Z axis direction.

If the active coordinate system is the GCS, the translational terms are in millimeters in the direction of the GCS axis. If the ACS is active, the units depend on the units/mm factor specified in the profile.

4.1.8 VIEw VN:

Changes the reference view assigned to the current window. If VN is not specified, the system will prompt the user again.

If the VN has not been defined before, the system will default it to a front view of the model.

4.1.9 Window WN:

Changes the current window to window number WN. If an invalid WN is specified, the system will ignore the request. If WN is not given, the system will prompt the user again.

Depending on the current window, this command will cause the entire screen to be redrawed. This happens when the requested window is not an active window number.

1) WIN 6

Results in a four window display format with

window 6 as the current window.

2) WIN 2

Results in a two window display format with window 2 as the current window.

4.1.10 CREate:

Invokes the Create Commands menu. See Create Commands for description.

4.1.11 PLOt:

Invokes the Plot Commands menu. See Plot Commands for description.

4.1.12 SAVe Dbase:

Saves the definition of the current model in the default diskette or disk device.

The system prompts the user for a file name to store the Data Base.

4.1.13 LOAd Dbase:

Loads a previously stored Data Base file.

The system prompts the user for the file name to load the Data Base from.

4.1.14 NEW Dbase:

Clears the current Data Base and screen and initializes default views.

The system request this function be confirmed or it will ignore the request.

4.1.15 Exit:

Exit the program without saving anything.

The system request this function be confirmed or it will ignore the request.

4.2 Create Commands:

The create commands interact with the user in order to define the model.

The following are the create commands supported by Silouett, only the capitalized portion of the command name needs to be specified to execute the command.

After creating the graphic entity, the system allows the user one chance to reject the just created entity before it is added to the data base.

4.2.1 ARC:

Allows the user to create an arc section anywhere in space.

The system requires the user to specify the coordinates of three points in space. The points can be specified either in a clockwise or counterclockwise direction. The extremes of the arc section are determined by the first and last point. The second point is a point on the arc.

4.2.2 CIRcle:

Allows the user to create a circle anywhere in space.

The system requires the user to specify the coordinates of three points in space. The points can be specified either in a clockwise or counterclockwise direction.

4.2.3 ELLipse:

Allows the user to create an elliptical section anywhere in space. An inplane rotation can be specified.

The system request input for:

1. Coordinates of center.
2. Beginning and End angles (°D) of elliptical section relative to the semi major axis.
3. Inplane rotation (°D).
4. Length of Semi major and Semi minor axis.

4.2.4 HYPerbola:

Allows the user to create a hyperbolic section anywhere in space. An inplane rotation can be specified.

The system request input for:

1. Coordinates of vertex.
2. Beginning and End angles (°D) of hyperbolic section relative to the semi transverse axis.
3. Length of Semi transverse and Semi conjugate axis.

4.2.5 PARabola:

Allows the user to create a parabolic section anywhere in space. An inplane rotation can be specified.

The system request input for:

1. Coordinates of vertex.

2. Beginning and End angles (°D) of parabolic section relative to the parabola axis.
3. Inplane rotation (°D).
4. Maximum and minimum values of the parabolic section.

4.2.6 POInt:

Allows the user to create a point in space.

The user is prompted for the coordinates of the point.

4.2.7 ATTributes:

Allows the user to change the current drawing attributes.

The number of attributes, ei.. number of colors, etc, depends on the system configuration. The following are the attributes supported at this time.

LColor (Line Color) Allows the user to change the current color used to draw lines and all

conic sections. Entities retain the color active at the time the individual entity was created.

LTYpe (Line Type) Allows the user to change the line type, ei.. solid, dash, points, center, etc.

LWIth (Line Width) Allows the user to specify the with of all lines created.

PCOlor (Point Color) Allows the user to specify the color of all points subsequently created.

PTYpe (Point Type) Allows the user to specify the symbol used to represent points, ei... cross, dot, etc.

SMOothness The value of this parameter determines the accuracy with which all conic sections are represented on the screen (not the data base). The value corresponds to the physical distance (mm) from the bisector to a chord on the critical radius of the curve and the curve itself.

4.2.8 QUIt:

Returns to the Global Commands menu.

4.2.9 LABt:

Returns to the previous menu.

4.3 Plot Commands:

The plot command allow the user to format the plot on the screen before it is actually plotted on a plot device such as an XY plotter.

4.3.1 CURrent VN:

Selects reference view VN as the current plot view. Multiple views can be active at any one time in the plot window.

4.3.2 PLOt views:

Plots all currently active plot views on the plot device.

4.3.3 REJect VN:

Allows the user to reject a previously selected plot view (VN).

4.3.4 RELocate VN:

Allows the user to relocate the specified view number (VN) anywhere on the plot surface.

4.3.5 RESet views:

Rejects all currently active plot views.

4.3.6 VIEw Select VN:

Allows the user to select the plot views.

4.3.7 QUIt:

Returns to the Global Commands menu.

4.3.8 LASt:

Returns to the previous menu.

APPENDIX B - EXAMPLE PLOTS

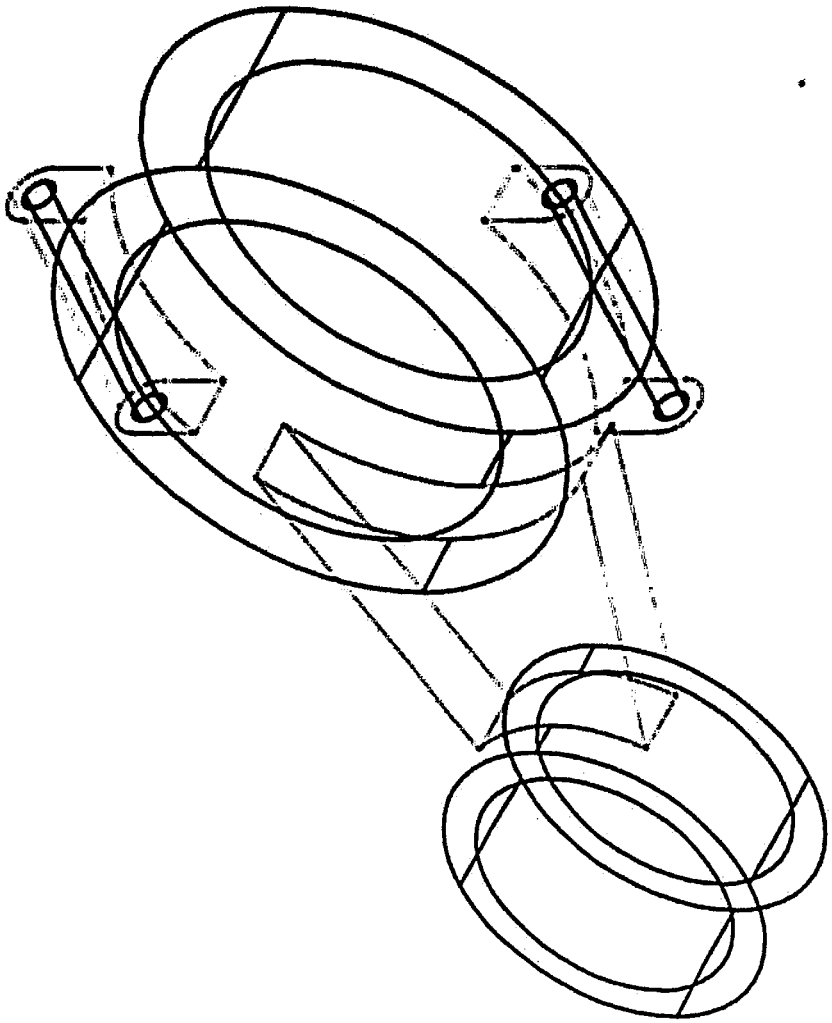


Figure 14. Connecting Rod - Isometric View

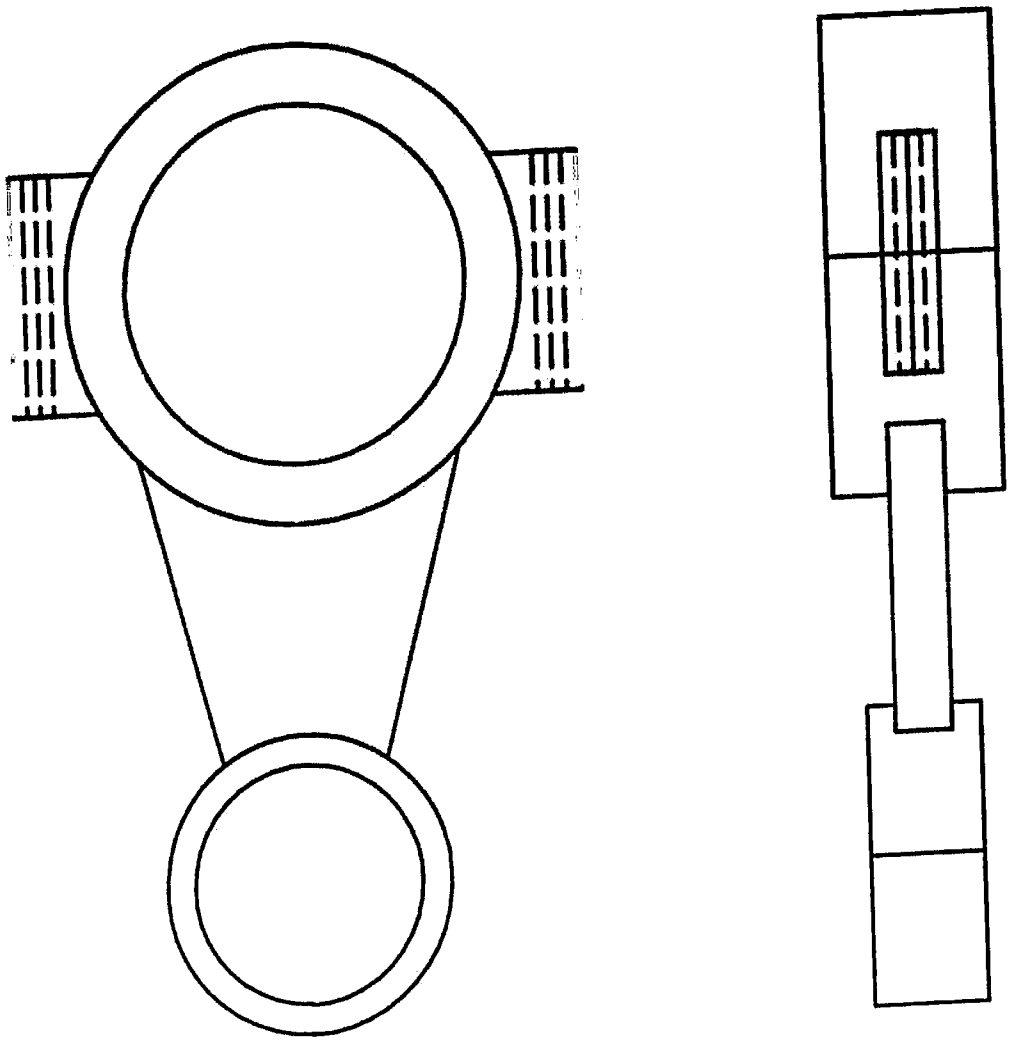


Figure 15. Connecting Rod - Front and Top Views

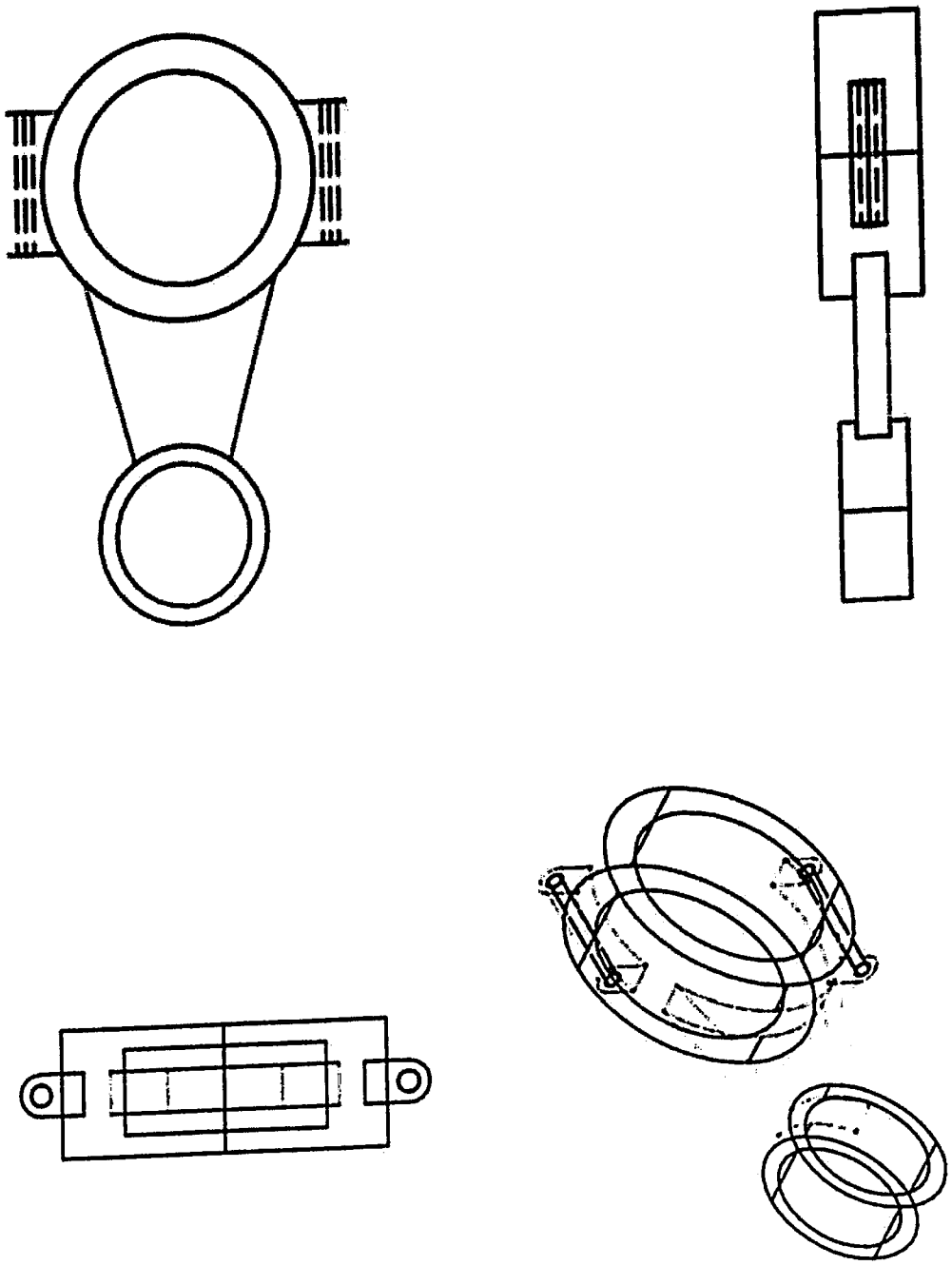


Figure 16. Connecting Rod - 4 View Display

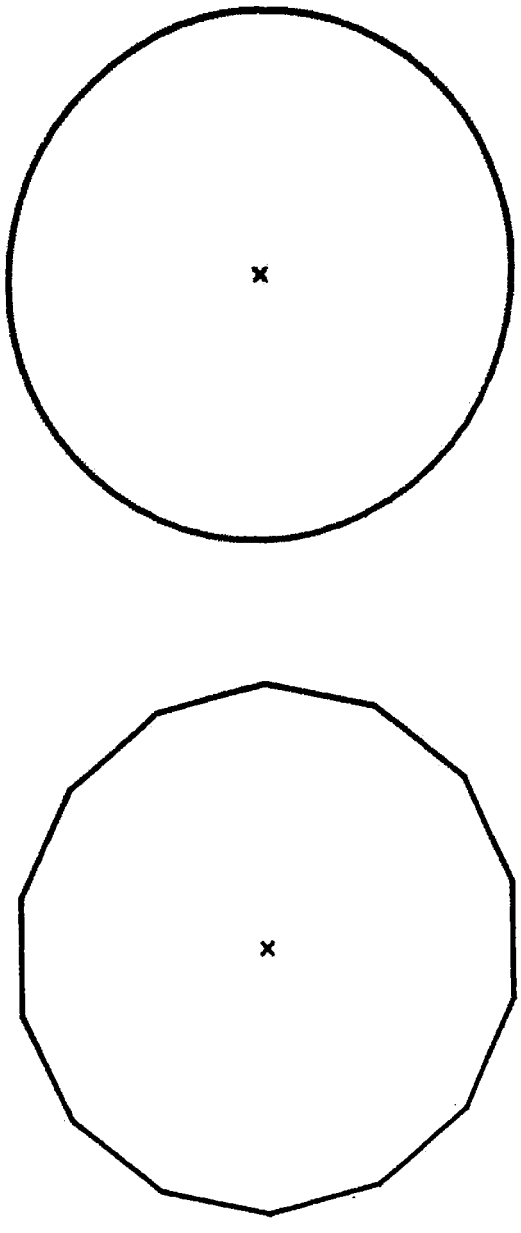


Figure 17. Circle - High and Low Resolution Drawing

APPENDIX C - DATA MODELS

Relational Data Model:

In this model data are represented in a two dimensional table called a relation. Data records are represented by rows and each column represents a record attribute. Within a relation, all records must be of the same type, therefore different entities are represented by different relations.

POINTS RELATION

ID	X	Y	Z
1	5	5	0
2	10	5	0
3	10	10	0
4	5	10	0

LINES RELATION

ID	X1	Y1	Z1	X2	Y2	Z2
1	5	5	0	10	5	0
2	10	5	0	10	10	0
3	10	10	0	5	10	0
4	5	10	0	5	5	0

Figure 18. Relational Table or Relation

The notion of using a Relational Data Model as an underlying structure of a DBMS was introduced in 1970 by Dr. E. F. Codd (IBM) [CODD70]. A true relational data base adheres to Codd's formal definition :

- No row can exactly duplicate any other row.
- There must be an entry in one column or a combination of columns that is unique for each row. This column or set of column is called a "key".
- There must be one and only one entry in each row-column cell, however the entry may be null or zero.
- The rows need not be in any particular order.

The last requirement is both a major advantage and major disadvantage of this model. It makes it easy to add new rows to the relation but searching for a specific entry may require a search of the entire relation.

1.1 Advantages:

- Simplicity - the end user is presented with a simple data model. Requests are formulated in terms of the information content.
- Non-Procedural Requests - because there is no positional dependency between the relations, requests do not have to reflect any preferred structure.
- Data Independence - the relational model removes the details of storage structure and access strategy from the user interface.
- Theoretical Foundation - the Relational Data Model is based in the well developed theory of relations, a characteristic only claimed by this model.

1.2 Disadvantages:

Although from a users point of view it is a simple model, the physical implementation is much more complex than for other structures. System performance remains the major disadvantage of this model today.

Hierarchical Data Model:

A Hierarchical Data Model is basically a hierarchical tree structure made up of nodes and branches. In this context , a node is a collection of data attributes describing the entity at that node.

The highest node of the hierarchical tree structure is called a ROOT. The dependent nodes are the lower levels in the tree. Every root node begins a logical data record, therefore the data base is made up of a number of trees. A hierarchical tree structure must satisfy the following conditions :

1. A Hierarchical Data Model always starts with a root node.
2. Every node consists of one or more attributes describing entities at that node.
3. The node on the preceding level is called a parent node while those that succeed it are called children nodes.
4. Every node occurring at one level has to be connected to one and only one parent node. Parent nodes can have multiple children

nodes.

5. Every node, except the root node, has to be accessed through its parent node.
6. There can be a number of occurrences of each node at each level. Each occurrence, except the root, has to be connected with a parent node occurrence.

2.1 Advantages:

- Proven by many existing data base systems
- Relatively simple to use and familiar to many data processing users.
- Performance can be predicted.

2.2 Disadvantage:

Not all relationships can be easily implemented, leading to redundancy in data storage.

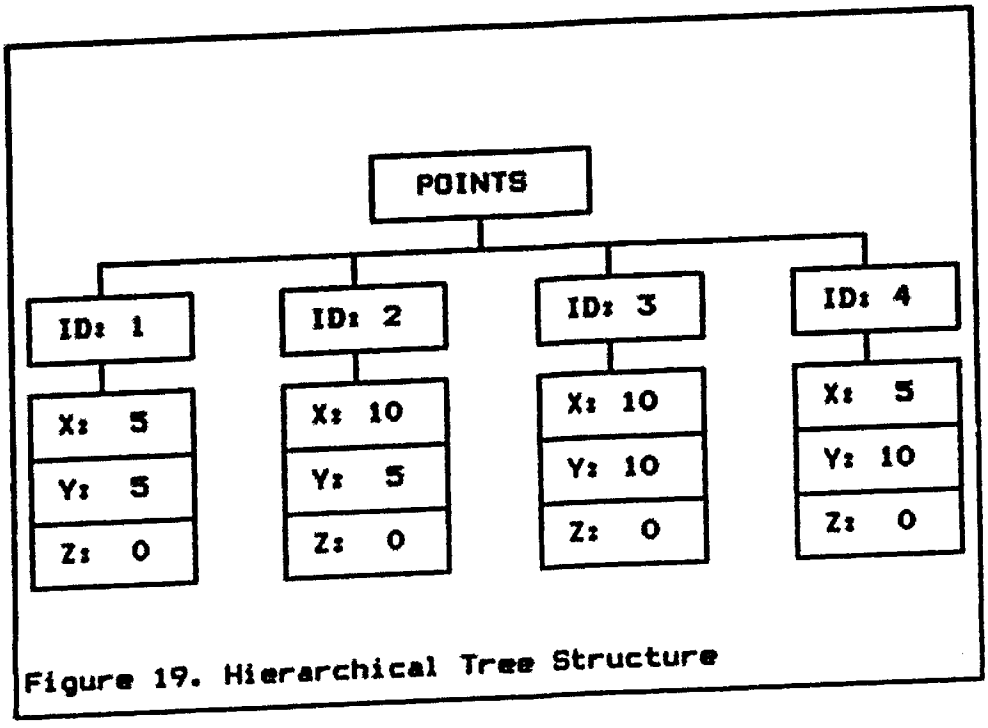


Figure 19. Hierarchical Tree Structure

Network Data Model:

The Network Data Model emerged from the 1969, 1971 and 1973 reports from the Conference on Data Systems Languages (CODASYL) and its Data Description Language Committee.

The Network Data Model interconnects the entities of a data base into a "network". Each record type is composed of zero, one, or more attributes (or data elements). A record type can have multiple "occurrences" in the data base.

Some characteristics of the Network Data Model include :

- Representation of one - to - many relationships. A owner record type can be a owner record type in a number of of set types, however, a record occurrence can not simultaneously belong to two or more owner record occurrences of the same set type. This special case of the Network Data Model is equivalent to the Hierarchical Data Model.

- Representation of many - to - many relationships can be implemented by creating two one - to - many relationships in a "Y" structure. That is, there would be two owners for the same member but each owner is of a different set type. Figure 20 on page 143 shows one of the advantages of allowing this kind of structure is the reduction of data redundancy, ie... the amount of data required to represent LINES is reduced by cross referencing the LINES entity to the POINTS entities.

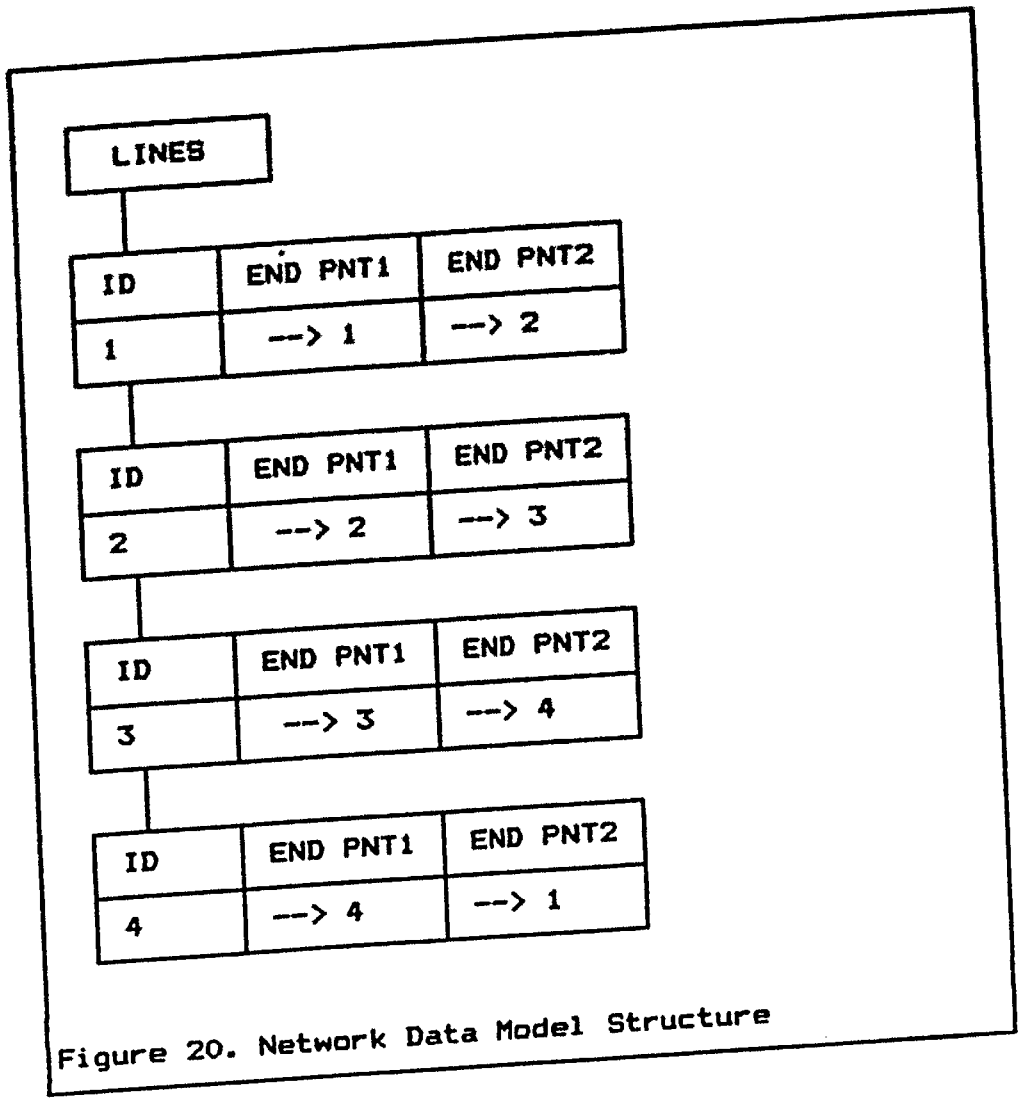


Figure 20. Network Data Model Structure

3.1 Advantages:

- The major advantage, as was the case with the Hierarchical Model, is the existence of many successful DBMS that use this structure.
- The ability to represent many - to - many relationships.

- Performance can be greatly enhanced by predefined data relationships.

3.2 Disadvantages:

The main disadvantage of this model is its complexity. The application programmer must be familiar with the data base structure in order to navigate through it in search of information.

When the data base is modified, great care must be exerted not to lose data independence. Since data can be cross referenced, it is easy to lose information when editing the data base structure.

VITA

The author was born in San Juan, Puerto Rico on January 23, 1953. His parents are Gerardo Sanchez Ferrer (deceased) and Lutgarda Sanchez De Las Casas.

He graduated from Colegio San Agustin (High School) in Madrid, Spain. After which he returned to Puerto Rico and attended the University of Puerto Rico where he received a Bachelor's of Science Degree in Mechanical Engineering Magna Cum Laude. He is a member of TAU BETA PI Engineering Honor Society and was in the Engineering honor roll while attending the University of Puerto Rico.

Upon graduation, he accepted a position with General Dynamics in San Diego, California. Most of his work at GD was related to the dynamic and thermal analysis of General Dynamic's F16 avionics.

In 1977, he accepted a position with his current employer, IBM Corporation. While at IBM, he has worked in several locations domestic and abroad. In his most recent assignment, was involved in the

design and development of a Personal Computer. At the completion of the requirements for the Masters Degree, the author will return to Boca Raton, Florida where he will continue working in the Exploratory Process Development area.

The author has a three year old daughter, Melissa Mariel.